Doctoral Dissertations 1896 - February 2014

1-1-1973

Elementary mathematics teacher training via a programming language.

Portia Clareon Elliott
University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_1

Recommended Citation

ELEMENTARY MATHEMATICS TEACHER TRAINING

VIA

A PROGRAMMING LANGUAGE

A Dissertation

By

Portia Clareon Elliott

Submitted to the Graduate School of the
University of Massachusetts in
Partial Fulfillment of the Requirements for the Degree of

DOCTOR CF EDUCATION

December 1973

Major Subject: Mathematics and Computer Education

ELEMENTARY MATHEMATICS TEACHER TRAINING

VIA

A PROGRAMMING LANGUAGE
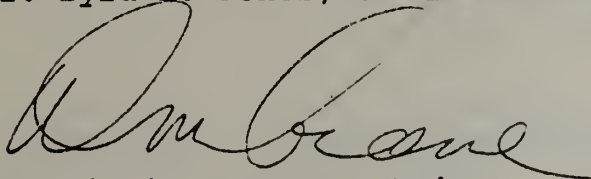

A Dissertation

By

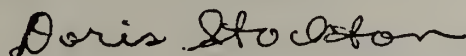Portia Clareon Elliott

Approved as to style and content by:
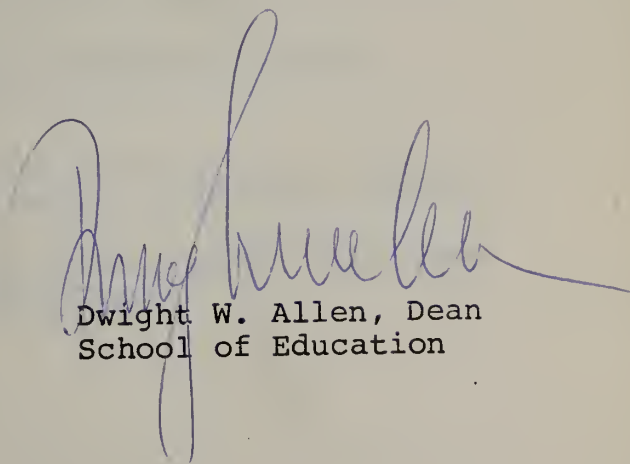
Dr. Byrd L. Jones, Chairman of Committee

Dr. William Greene, Member

Dr. Howard A. Peelle, Member

Dr. Doris Stockton, Member

Dwight W. Allen, Dean
School of Education

December, 1973

This work is dedicated with love to

my father, Frank M. Elliott, a biology teacher,
who gave me his love for teaching

my mother, Erma D. Elliott, a mathematics teacher,
who gave me her love of mathematics

my sister, Verneva, a fifth grade teacher,
who gave me this problem to tackle

my sister, Eleanor, a first grade teacher,
who gave me the loan of her artistic talents

my sister, Cynthia, an English teacher,
who gave me encouragement and editorial comments

my sister, Rebera, the soon to be M.D.,
who gave me "free" medical advise to keep up my strength

my brother, Thomas, the college student,
who gave me criticism, laughter and love, in that order

and finally, to my whole family, whose quest for
knowledge, whose dedication to self-fulfillment,
and whose faith in the "next" generation made
all this possible.

ACKNOWLEDGEMENTS

It would be a feat comparable to disproving
" Godel's Incompleteness Theorem" or solving the famous
"Halting Problems" of automata theory to try to "effectively
enumerate" all the persons who have given this author
encouragement, suggestions, constructive criticisms and
support during the writing of this document. No attempt,
therefore, will be made at such an enumeration, but a
subset of this list will be given.

First, I would like to thank my dissertation
committee in the persons of Dr. Byrd L. Jones, Dr. William
Greene, Dr. Howard A. Peelle, and Dr. Doris Stockton for
services beyond all expectations.

I would also like to thank the members of my
advisory sounding board who were critical at times, but
constructively so.

I would like to express appreciation for the
administrative assistance of Spike Paranya and Jack Fagan
who included the MAP Course in their Methods Potpourri
offerings and for the cooperation of the "U-MASS" Computing
Center in this venture.

"EVENTUALLY, PROGRAMMING ITSELF WILL
BECOME MORE IMPORTANT EVEN THAN
MATHEMATICS IN EARLY EDUCATION."

Marvin Minsky

ELEMENTARY MATHEMATICS TEACHER

TRAINING VIA

A PROGRAMMING LANGUAGE (DECEMBER, 1973)

Portia C. Elliott, B.A., Fisk University

M.A., University of Michigan

Directed by: Dr. Byrd L. Jones

Traditionally, mathematicians, mathematics educators, teacher supervisors and teachers themselves have pointed to the woeful inadequacies of elementary school teachers' mathematics abilities and training. For this reason many elementary school teachers join the ranks of the "disadvantaged" when they are placed unprepared in "unfavorable positions"--namely, in modern elementary mathematics classes.

To achieve a reversal in the condition of "disadvantageousness," teachers must become knowledgeable of modern mathematics content, as well as methodologies and they must learn to become facilitators in educational environments of the future.

To achieve these objectives it will be necessary to revitalize many existing mathematics teacher training programs by integrating methods and content and by incorporating vital and pervasive educational technology into teacher training offerings.

This dissertation explores the feasibility of using a computer as "an obedient student" in teacher training programs and it explores the possibilities of using A Programming Language (APL) to clarify selected topics in mathematics.

The specific purpose of this manuscript is three-fold: 1) to develop curriculum materials in mathematics education which utilizes computer technology and A Programming Language; 2) to strengthen basic mathematics understandings of prospective elementary school teachers by having them examine, modify, and write "glass box" computer programs; and 3) to introduce innovative methods for teaching elementary school mathematics topics.

The central theses which give direction to this document are:

1. A programming language such as APL is well-suited as a conceptual framework for teaching mathematics concepts to elementary school teachers.
2. Programming heuristics are well-suited as "epistemological primitives" for mathematics teacher training courses.
3. A computer, in its role as "obedient servant" is the ideal pupil to help teachers clarify and organize their own thoughts and build self-confidences in mathematics and programming.

The dissertation is divided into five chapters. Chapter I discusses the problem area and suggests recommendations for revitalizing teacher training programs. Chapter II reviews the literature at the point of intersection of

mathematics education, teacher training and computer
education. Chapter III describes and evaluates the
"Mathematics and Programming" (MAP) course taught by this
author to prospective elementary school teachers at the
University of Massachusetts. Chapter IV presents the
materials developed by this author for use in the MAP
course--"APL for Teacher-Learners" and "A Supplemental
Guide to APL for Teacher-Learners." And finally, Chapter V
gives a summary, conclusions and recommendations for future
research in the areas of computer-augmented-inquiry (CAI)
and computer-augmented-teacher-training (CATT).

TABLE OF CONTENTS

C H A P T E R   I

INTRODUCTION

The Disadvantaged Teacher

The word "disadvantaged" has been bandied about the
educational arena for the past decade. More often than not,
the word has been used to describe children of low socio-
economic backgrounds who populate inner-city elementary
schools. "Disadvantageousness" is not just a condition
symptomatic of inner-city students; it is also an adequate
descriptor of a large majority of the elementary school
mathematics teachers.

If a teacher is offended by this labeling, it may be
due to the negative connotations harbored for the word
"disadvantaged." But, if this indeed is the case, perhaps
one should refrain from using this word to describe children.
If "disadvantaged" is taken to mean "being in an unfavorable
situation," not an irreversible one, mind you, it should
not evoke undue criticism.

"Disadvantaged" elementary school teachers are not
difficult to spot by laymen; nor have they made their
presence obscure to academicians. Laymen need only observe
traditional elementary school mathematics classes, where

students and teachers are often prisoners of fixed curricula, out-moded methods, and negative "mathematics attitudes." Or, laymen can listen to comments made by some elementary school teachers who readily confess: "I've always hated mathematics;" "Mathematics was my worst subject;" or "I am just not mathematically minded."

Academicians have traditionally pointed to the woeful inadequacy of elementary school teachers' mathematics ability and training. Educational researcher Emma Carrol, for one, in a study of pre-service teachers revealed that education students did not have the mathematics understanding needed to teach arithmetic competently.[1] Another study by Marian Wozencraft revealed that although the elementary education students tested were at the 65th percentile on the American Council of Educational Psychology Test, they only achieved at the 7th grade median on a standardized mathematics test.[2] Reviewers Jack Sparks[3], Jacob Orleans and Edwin Wandt[4] reported that mathematics competence of

---

[1]Emma Carroll, "A Study of the Mathematical Understanding Possessed by Undergraduate Students Majoring in Elementary Education," (unpublished Ed.D. dissertation, Wayne State University, 1961).

[2]Marian Wozencraft, "Even the Teacher Can't Do It!," Journal of Teacher Education, XI (September, 1960), 387-390.

[3]Jack Sparks, "Arithmetic Understanding Needed by Elementary School Teachers," The Arithmetic Teacher, VIII (December, 1961), 395-403.

[4]Jacob Orleans and Edwin Wandt, "The Understanding of Arithmetic Possessed by Teachers," Elementary School Journal, LIII (May, 1953), 501-507.

in-service teachers was no greater than that of pre-service teachers.

A look at the relative percentages of enrollment in mathematics content courses offered at the university level gives more documentation that teachers of elementary school mathematics are indeed "disadvantaged." Richard Hunkler sampled 211 sixth grade teachers to determine whether or not these representatives of seventeen colleges had met the Committee for Undergraduate Preparation in Mathematics' (CUPM) recommendations of a minimum of twelve semester hours of mathematics. His findings indicated that only one of the 211 had satisfied minimum requirements. Ninety percent had less than six hours of mathematics and sixty percent had not completed any hours.[5]

"Disadvantaged" elementary school mathematics teachers have not escaped the critical eye of university professors. John F. LeBlanc, professor of mathematics at Indiana University, stated that,

> . . . the typical elementary school teacher
> of 1970 has been prepared as a general elementary
> school teacher. She may have had no other prep-
> aration to teach mathematics than a methods course;
> at worst, only a few weeks in a general methods
> course; at best, a couple of mathematics courses

---

[5]Richard Hunkler, "A New Look at the Implementation of the CUPM Level I Recommendations," School Science and Mathematics, LXIX (May, 1961), 423-425.

and a three-credit course in methods . . . ordinarily
she does not have a clear picture or understanding
of what she is trying to do in mathematics.[6]

The assessment that elementary school teachers lack
an understanding of mathematics and are being inadequately
prepared for the elementary school classroom was confirmed
by Andre Brousseau, professor at Center College of Kentucky.
He amplified LeBlanc's position by saying, "many elementary
school teachers are suffering further from a lack of
appreciation of the mathematics concepts they are required
to teach."[7]

Responsibility for inadequately prepared elementary
mathematics teachers must be shouldered in part, by mathe-
matics teacher educators, mathematics program directors,
supervisors, and, of course, the teachers themselves.
Educators sampled from each of these categories all
diagnosed deficiencies in mathematics teacher training at
the elementary level. Their concerns were captured in the
following questions which they submitted in response to a

---

[6]John LeBlanc, "Pedagogy in Elementary Mathematics--
Time for a Change," The Arithmetic Teacher, XVII (November,
1970), 605-609. N.B. Professor LeBlanc has not avoided the
sexist policy of calling all elementary teachers she, while
reserving the pronoun he for other teachers.

[7]Francis J. Mueller, ed., "Forum on Teacher
Preparation," The Arithmetic Teacher, XVIII (April, 1971),
265-267.

survey conducted by Larry Hatfield, commissioner of

Teacher Education of the National Council of Teachers of

Mathematics:

> What can be done about the mathematics
> deficiencies and/or incompetencies of beginning
> elementary education majors?
> How do we get elementary teachers in the
> field who, having had content courses of the
> CUPM variety and "methods" courses, are still
> largely ignorant of actual content of elementary
> school mathematics?
>> a. Where do these training programs
>> fail us?
>> b. What changes in programs can we
>> propose to remedy this situation?
> Are content courses in mathematics for
> elementary teachers doing their job--or should
> we be re-examining the philosophy that college
> mathematics courses make better elementary
> teachers? Should we consider the alternative
> of combined content and methods courses--or
> some other alternative?[8]

All of these expressed concerns, coupled with

teachers' own private admissions, strongly suggest that

elementary school teachers are "disadvantaged." They have

been placed unprepared in unfavorable situations——namely,

in elementary school mathematics classrooms.

Revitalization of Teacher Training Programs

How we achieve a reversal of this unfavorable

condition imposed on many elementary school teachers

becomes a crucial issue. With the problem area defined,

---

[8]Larry Hatfield, "Some Issues and Problems in
Mathematics Teacher Education," Papers from the Forum on
Mathematics Teacher Education NCTM (paper presented at the
National Council of Teachers of Mathematics meeting, Chicago,
Illinois, April, 1973), pp. 1-7.

we can now turn our attention to the needs in elementary mathematics teacher training.

Generally what is needed are revitalized mathematics teacher training programs. Any new program must emphasize improved mathematics competencies in content areas while strengthening teachers' pedogogical strategies. Any new program must give teachers confidence to teach existing curricula while providing them with skills in dealing with current technological innovations in education. And finally, any new program must encompass knowledge of cognitive and affective development in children while stressing improved self-concepts of students and teachers alike.

To achieve these goals, it is not necessary to abandon established teacher training mathematics programs. Rather, what is needed are new strategies to strengthen the link between methods and content components, new strategies to enrich curriculum, and new strategies to encompass current technology and affective approaches in existing programs.

The following chapters explore the use of the computer and a programming language as an indispensible tool for achieving the link between methods and content components and for meeting the need to revitalize mathematics programs. Specifically, the materials contained in this document will attempt to suggest ways the computer can

be used in elementary mathematics teacher training courses
to improve teacher competencies, understandings, pedagogical
strategies, and self-concepts.

Computers in Teacher Training

The introduction of computers in teacher training
mathematics programs has been strongly recommended in
numerous council and commission reports. The Conference
Board of the Mathematical Science Committee on Computer
Education reported:

> . . . While there have been widespread
> improvements in the education of prospective
> teachers, training in the use of up-to-date
> technology is still not included in the
> curricula of most teacher training
> institutions . . .
> . . . The computer is a valuable teaching
> tool, but it will be effective only if
> teachers are trained in its uses. Moreover,
> it is the teacher, properly trained, who can
> convey to students the power of this tool
> and how to use it . . . [9]

This committee also recommended that the National
Science Foundation sponsor the development of basic
curriculum to train teachers in the use of computers.

Peter J. Hilton, in an article at the National
Council of Teachers of Mathematics (NCTM) Forum suggested
the use of computers in teacher training.

---

[9]Conference Board of the Mathematical Sciences
Committee on Computer Education, Recommendations Regarding
Computers in High School Education, Washington, D.C.
(April, 1972), p. 14.

He wrote:

> It is not necessary to say too much about
> the increased role of the computer in so many
> aspects of science and commerce. The importance
> of understanding the way in which a computer
> functions leads to the conclusion that notions
> of algorithms and flow diagrams should be
> introduced extremely early in the child's education.
> It follows, therefore, that the elementary school
> teacher must understand the basic principles
> governing the operation of the computer.[10]

Beyond the crucial need for understanding computers

per se, Hilton also described the benefits of increased

understanding for certain facts of mathematics, when taught

in the context of computer programs:

> There is . . . the tremendous advantage to be
> gained from being able rapidly to check the
> validity of a procedure which one has devised
> to solve the problem. Thus, not only would an
> increased emphasis on the computer better
> prepare the child for the modern world, it
> should also actually render his experience
> of mathematics more effective.[11]

The committee on Guidelines of the Commission on

Pre-Service Teacher Education of the NCTM proposed adding

the following objectives to the "minimal knowledge and

competency list for elementary school mathematics teachers."

The committee wrote that teachers should:

---

[10]Peter J. Hilton, "Recommendations on Course
Content for the Training of Teachers of Mathematics,"
Papers from the Forum on Mathematics Teacher Education
NCTM (paper presented at the National Council of Teachers
of Mathematics, Chicago, Illinois, April, 1972), pp. 13-14.

[11]Ibid., p. 14.

be able to develop new algorithms for
operations and be able to test the
effectiveness and correctness of algorithms . . .
be able to use at least one computer language
(e.g. Fortran IV, BASIC, etc.) sufficiently
well so as to be able to solve problems of
appropriate level and complexity with the
aid of a modern computer.[12]

In Larry Hatfield's review of computer-extended

mathematics programs, he recommended taking teachers into

consideration when preparing materials for computer

mathematics:

Ideally, teachers need to know, to some
specified degree of confidence, when to teach
and use programming, for which types of students,
for which content selections, and how this should
be done (which programming, what kind of computer
access, which combinations of classroom pedagogies,
etc.). And throughouut [sic] all such specifica-
tions, the intended learning outcomes should be
clearly explicated with performance terms and in
a taxonomic specification...(Furthermore, studies
of mathematical learning in any context should
employ existing theories of learning and child
development in their planning, and then eventually
feed back empirical results which support or lead
to modifications of these theories.)[13]

The use of the computer in mathematics teacher

training, then, has promise for three-fold impact: (1)

familiarizing future teachers with a vital and pervasive

technology; (2) strengthening basic mathematics concepts

---

[12]Committee on Guidelines of the Commission on
Pre-Service Teacher Education of the NCTM, Paper from the
Forum on Mathematics Teacher Education NCTM (paper
presented at the National Council of Teachers of Mathematics,
Chicago, Illinois, April, 1972), pp. 26-28.

[13]Larry L. Hatfield, "Computer-Extended Problem
Solving and Enquiry," (paper presented at the Annual NCTM
Meeting, Anaheim, California, April, 1971), p. 28.

through programming; and (3) introducing innovative methods for elementary mathematics teaching.

The specific purpose of this document, therefore, is to describe curriculum materials developed to teach elementary school teachers concepts in mathematics via A Programming Language; to suggest pedagogical strategies for using computers to teach elementary mathematics concepts; and to explicate the cognitive and affective behavioral objectives to be stressed when computer-related programming materials are employed in elementary school settings.

The central theses which give direction to this document are:

1. A programming language such as APL[14] is well-suited as a conceptual framework for teaching mathematics concepts to elementary school teachers.
2. Programming heuristics are well-suited as "epistemological primitives"[15] for mathematics teacher training courses.
3. A computer, in its role as "obedient servant" is the ideal pupil to help teachers clarify and organize their own thoughts and build self-confidence in mathematics and programming.

---

[14]APL (A Programming Language) is a general-purpose interactive computer programming language developed by Kenneth Iverson (1962) and supported by IBM. Originally conceived as a unifying mathematical notation, APL is a language with simple rules and yet offers the user great computational power and flexibility.

[15]Seymour Papert, "Teaching Children to be Mathematicians versus Teaching About Mathematics," New Educational Technology (Spring, 1973), 18.

Learning by Teaching and Learning Begets Learning

The rationale for this work is based on two assumptions. (1) "A good way to learn something is to teach it," and (2) "Learning begets learning." The first assumption clearly suggests cognitive gains as well as pedagogical possibilities inherent in the use of computers as "obedient students." The second assumption hints at affective consequences inherent in having teachers and students as partners in the learning process.

"The best way to learn something is to teach it"

An immediate consequence of casting computers in the role of "obedient students" may be concept clarification without fear of contributing to confusions of real-life students. How many times have we as teachers found ourselves in the middle of an unorganized lecture when a fuzzy concept is finally made clear to us. This is a great revelation for the teacher, but, often times, the "aha" is not shared by the curious, now confused, students. Using the computer as an "objective reflector of our own understandings,"[16] could maximize the spur-of-the-moment revelations and minimize confusions resulting from illogically presented materials.

---

[16]Howard A. Peelle, "Pygmalion's Computer," Controversies in Education (Philadelphia: W.B. Saunders), to be published 1974.

Seymour Papert, co-director of the Artificial Intelligence Laboratory at Massachusetts Institute of Technology, has expressed the belief that persons with fuzzy notions about mathematics concepts will have these notions clarified if they articulate their thoughts by writing and revising computer programs. The active process of writing a program, Papert felt, forces one to consider possible misunderstandings and ambiguities in a discourse.[17]

Another immediate consequence of using computers-as-pupils may be the acquisition of teaching techniques which may be employed in actual classroom situations.

Besides the observable teaching strategies employed during a course using computers and novice programmers, the heuristics involved in the actual programming process suggest strategies for teaching that heretofore have not been high-lighted in elementary schools. For example, emphasis on breaking difficult tasks down into manageable procedures and subprocedures is a valuable heuristic for students to use.[18] The concepts of bugs and debugging provide valuable lessons in problem solving. The student should look at errors as

---

[17]Seymour Papert, "Teaching Children Thinking," New Educational Technology (Spring, 1973), 3.

[18]Papert, "Teaching Children to be Mathematicians Versus Teaching About Mathematics," 19.

"bugs," to be traced down, conquered or tamed, rather than as reflections of their own inabilities.

Learning by teaching a computer may have far-reaching implications. Increased computer literacy may help dispel some of the myths currently surrounding computer technology. For laymen and educators alike, a mystique has surrounded computers--suggesting their complexity, their omnipotence, or their artificial intelligence. Prospective teachers who develop literacy for computers and programming, it is hoped, will gain a respect for the power of this new technology as well as a knowledge of its real uses and its limitations.

These potential consequences of casting computers as pupils may ultimately give teachers new confidences in mathematics, in programming, and in teaching. These consequences may also help them break out of the "teachers teach as they were taught cycle" and begin new cycles of excellent teaching and positive attitudes about mathematics and computers.

"Learning Begets Learning"

The learning begets learning maxim has interesting and immediate implications for teachers and students alike.

---

19 Seymour Papert, "A Computer Laboratory for Elementary School," New Educational Technology (Spring, 1973), 27.

Prospective teachers who are cast in the role of learners--
as they are in a "Mathematics and Programming" (MAP) Course
(Described in Chapter 3)--may initially feel the awkwardness
of their role. They will be given a new language to master
and old concepts to reconsider. This will probably place
them in a state of disequilibrium[20] which, if negotiated
properly, may make them more empathetic, more sensitive
teachers. For teachers this is the first step in the
learning-begets-learning spiral--namely, learning how
learners learn. Educators have used this paradigm to teach
teachers how to teach reading. Dr. Paul McKee, consultant
for Denver Public Schools and author of Reading: A Program
of Instruction for the Elementary Schools and A Primer for
Parents, strongly urged that parents and elementary school
teachers put themselves in the place of first graders to gain
insight into the reading process. He wrote:

> . . . if you could literally put yourself in
> the place of the first-grade child who is, for
> the first time trying to learn to read . . .
> then you would be able to see what he's up
> against . . . You would be able to avoid many
> of the errors all of us make . . . [21]

---

[20]Herbert Ginsburg and Sylvia Opper in Piaget's
Theory of Intellectual Development (Englewood Cliffs, N.J.:
Prentice-Hall, 1969), pp. 172-179, discussed Piaget's
notion of "equilibration" in reference to assimilation and
accommodation that goes on in a cognitive system of a child.

[21]Paul McKee, A Primer for Parents (Boston:
Houghton Mifflin Company, 1964), p. 4.

This pristine state for learning a discipline cannot be wholly recreated for teachers or prospective teachers because of their years of "experience" at learning. But, if they are willing to temporarily abandon the perspectives derived from the "experience" in favor of disequilibrium and insecurity, they might reap rewards-- new sensitivities and empathy for learners.

Sensitizing teachers is only part of the total learning begets learning process. The learner and his learning experience must now be taken into consideration. The learning experience must be open-ended if the full potential of learner is to be actualized; the learning experience must be success-oriented if the self-confidence of the learner is to be built; but, most important, if learning is to truly beget more learning, the learner must be given control over his environment, as well as concomitant responsibility for his own learning.

The computer may be a valuable tool in establishing these open-ended, success-oriented, student-controlled, prerequisites for the upward spiral of learning. This type of use of the computer is, as Howard A. Peelle, Director of the Instructional Applications of Computer Program at the University of Massachusetts wrote, "180° from the kind of CAI characterized by lock-step tutorial, drill-and-test instructional sequences." To children and teachers who

typically have been "powerless in authoritarian educational systems" use of the computer in this manner potentially enhances self-concepts and "opens up new worlds of learning-- active learning, learning with <u>power</u>." 22

There may also be far-reaching implications of the learning begets learning spiral. For students, with enhanced self-concepts, new self-confidences, and histories of learning success behind them, this may be the beginning of an upward spiral of learning. This continuous learning process may have impacts on the society-at-large.

For teachers, this maxim may mean increased responsibility to ensure that student learning spirals do not exacerbate alienation or elitism. Indeed, learners may feel powerful with increased competences in this new technology, but as facilitators in the learning process we must help students turn their feelings of power and control into constructive channels--namely, into channel of <u>self-control</u>.

Daniel Jordan, Director of the ANISA Project at the University of Massachusetts, stated that we must facilitate the actualization of human potential in ways that create further potential (further learning) and which guarantee

---

22 Howard A. Peelle, "The Computer Glass Box: Teaching Children Concepts With A Programming Language," to be published by <u>Educational Technology</u> (Spring, 1974). (Italics in text.)

the emergence of self-images or identities which enable humans to take charge of their own destinies.[23] He and co-author Donald Streets further warned:

> Whatever the actualization process, it must safeguard the human rights of all men and enable them to assume social responsibilities which maintain a social system and environment that supports the production and peaceful growth of humankind.[24]

In conclusion, the curriculum materials developed in this document are based on the principle that "knowledge is not enough." Knowledge must be accompanied by humane ways of using it. Besides cognitive considerations embodies in the materials "APL for Teacher-Learners," (See Chapter IV) affective aims are suggested in the teacher's guide (See Chapter IV) which accompanies the manual.

The ultimate success or failure of these materials, of mathematics teacher training programs, and indeed, of a society making productive use of computers rests in the abilities of teachers and learners to achieve the right balance between the cognitive and the affective domains of development. Computer literacy for humanity must be our watchword.

---

23 Daniel C. Jordan and Donald T. Streets, Releasing the Potentialities of the Child: A New Perspective on Child-Rearing, Day Care and Early Childhood Education (an unpublished paper written at the University of Massachusetts describing the ANISA Model) (November, 1972), pp. 168-169.

24 Ibid.

C H A P T E R   I I

REVIEW OF LITERATURE

Introduction

Potentials for using technology in education are virtually unlimited. Yet there are those educators who would attempt to ignore these possibilities or sheer away from them. This posture, it would seem, is a mistake. Educators should be about the task of interpreting the technology in educational frameworks. The following review examines the literature on the educational possibilities inherent in computer technology.

The entire "educational computer" dragon will not be taken to task in these pages. For it seems that as fast as a section is lopped off, a new part is generated. Rather, areas which relate to the research implications of this investigation (mathematics education/ and teacher training) will be examined. This review of related research will weigh first the field of Computer-Assisted-Instruction (CAI)[1] and review those prototypic projects with mathematics and/or teacher training components.

---

[1] CAI--Computer-Assisted-Instruction is a method relying on a computer to present specified material in any of a variety of modes and styles to a number of students individually for the purpose of learning.

Attention will then be given to the field which can be called Computer-Augmented-Inquiry (CAI_1)[2]. The section will be concluded by comments on Computer-Augmented-Teacher-Training (CATT)[3] which is a direction just recently embraced by educators.

Computer-Assisted-Instruction Reviewed

In the early nineteen sixties, the "promethean gift"[4] of computer technology found its way into many elementary, secondary, and college classrooms. The gift was hailed as a godsend by those who foresaw its potential for individualizing instruction, while others pessimistically warned of "Pandora's dowry" yielding: "impersonalization,"[5] "dehumanization,"[6] and "subjugation."[7]

Let us examine this "gift." In the forefront of the field of CAI we find men like Patrick Suppes at Stanford,

---

[2]Computer-Augmented-Inquiry refers to use of computers as "partners" not "masters" in instructional investigations.

[3]Computer-Augmented-Teacher-Training was coined by the author for this investigation.

[4]Prometheus was god of fire in Greek Mythology - forefather of modern technological society.

[5]Patrick Suppes, "Computer Technology and the Future of Education," Phi Delta Kappan, April, 1968, p. 422.

[6]Dwight Allen and D.D. Bushell, ed., "Computers: Like Drugs and Atomic Power," Phi Delta Kappan, April, 1968, p. 464.

[7]Arthur W. Luehrmann, "Large Scale CAI -- The NSF Program," Sigcue Bulletin: Computer Uses in Education, Vol. 6, No. 3 (June, 1972), 10.

Donald Bitzer at the University of Illinois, and Kenneth Stetten of the Mitre Corporation, all of whom acknowledge the potential evils of poorly administered CAI, but believe in the great goods of computerized instruction.

Patrick Suppes, director of the Institute for Mathematics Studies in the Social Sciences at Stanford University, heads one of the largest CAI projects in the nation. This project reportedly uses drill-and-practice techniques to teach numerous topics to elementary school students. Mathematics and reading have been taught using these techniques since the project's inception in 1966.[8]

Bitzer's PLATO IV (Programmed Logic for Automatic Teaching Operations) project is one of the most ambitious CAI systems ever attempted. Pre-programmed materials for grade school, community college, and university students are written by individual teachers with help from the PLATO IV staff. Several hundred self-contained lessons, including drill-and-practice work in mathematics, are stored on the system to be administered to students upon a teacher's request.[9]

---

[8]Robert F. Bundy, "Computer-Assisted Instruction -- Where Are We?," Phi Delta Kappan, April, 1968, p. 428.

[9] Allen L. Hammond, "Computer-Assisted Instruction: Two Major Demonstrations," Science, CLXXVI (June, 1972), 1110-1112.

The TICCIT (Time-Shared Interactive Computer-Controlled Information Television) system has the "explicit goal of showing that effective CAI can be produced, packaged and delivered economically . . . " by computer-controlled TV. Pre-packaged lessons in mathematics as well as other disciplines have been developed by TICCIT staffers.[10]

The three projects previously reviewed are prototypes of projects which have implemented CAI more as "Computer-Administered Instruction" and will be evaluated by Educational Testing Services (ETS) in 1974 for educational cost-effectiveness. Student and teacher control in these projects is limited. Students have some control over work sequences, over auxiliary materials they will use, and when they will start and stop. Teachers, in the same vein, have some authority over which "pre-packaged" materials they will give to students, and in some cases, they control the amount of time students will spend at the terminals. Lessons are doled out to students on a reward-punishment basis--if the sequence is correct, branch to next sequence; if incorrect, try again. And, rigid roles are prescribed for teachers using this computerized material.

There are those educators, in the ranks, who have looked at the beautifully wrapped "computer-administered instruction" "gift" and found flaws in its packaging.

---

[10]Ibid.

Those closely criticizing these systems express fear of narrowly construed CAI, fear of projects overshadowing other educational values, and fear of too much "computer-control." Even though much dissention is expressed, few dissenters are willing to advocate total abandonment of these computer-administered projects.

Certain kinds of information may be better suited to this administering of instruction, but opponents feel that there may be more creative uses of computers than mere ".task-masters."

Paul Berry et al. wrote:

> . . . Much of the work in educational computing has been done by people who narrowly construe "computer-aided instruction" as an extension of programmed instruction, and the computer as a successor to the Skinnerian teaching machine.[11]

Arthur W. Luehrmann expressed his concern for colossal projects which suggest subjugation of students in the name of teaching when he stated:

> What perturbs me about PLATO and MITRE projects is that, whether failures or successes, they will be big failures or big successes--so big that they may will eclipse other universes of educational computer use that are of deeper significance and value.[12]

---

[11]Paul Berry, A.D. Falkoff, and Kenneth Iverson, "Using the Computer to Compute: A Direct but Neglected Approach to Teaching Mathematics," IBM Technical Report No. 320-2988, New York Scientific Center (May, 1970), 1.

[12]Luehrmann, "Large Scale CAI--The NSF Program," 10-11.

To prevent eclipse of other educational computer possibilities by computer-controlled CAI, K. Iverson and P. Berry suggest a new style of computer use. They see computers as "arbiters of inquiries" rather than "administers of instruction." They wrote:

> This style of use differs sharply from most of present-day instructional computing. In particular the computer does not itself initiate any of the dialogue, but serves only to report the results of what are in effect mathematical experiments proposed by the class. There is no program in control . . . [13]

Seymour Papert had this to say about contemporary CAI projects and their efforts to control students and "rote-ized" material:

> . . . If you consider what is happening in computers in education . . . you will find first of all that a lot of it is quite clearly in the category of using these fantastic powerful machines to dish out the same old trash in a thinly disguised version of the same old way. Except it's rather worse, because you've chosen those parts that can be most easily rote-ized.[14]

Computer-Augmented Inquiry Reviewed

The disquietude of the last four educators suggests at least some room for revitalization and reexamination of CAI.

Berry, et al. "Using the Computer to Compute," 18.

Seymour Papert, "Student and Computers--Toward A Deeper Involvement," Sigcue Bulletin: Computer Uses in Education, Vol. 6, No. 3 (June, 1972), 15.

More importantly, their dissent is at the core of the philosophy for Computer-Augmented Inquiry which is to turn power and control of computers over to the user.

"Programming As A Conceptual Framework For Teaching Mathematics"[15]

The earliest developments in CAI_1 were understandably in the area of mathematics. The principle of algorithms on which computing is based forms the framework for teaching mathematics principles.

Wallace Feurzeig, _et al._ advanced the general thesis that:

> . . . mathematics could be developed and presented in the framework of programs and . . . this kind of presentation would greatly enhance teaching and learning.[16]

Using the LOGO[17] programming language, this thesis has been tested in the areas of geometry, logic and arithmetic in both the elementary and junior high school mathematics classes with impressive results.

---

[15]Wallace Feurzeig, _et al_. "Programming-Languages as a Conceptual Framework for Teaching Mathematics," _NSF Report_, Bolt, Beranek and Newman, Inc. (June, 1971).

[16]_Ibid_., p. i.

[17]LOGO is a programming language developed by Bolt, Beranek and Newman, Inc. specifically for use in teaching.

Edwin Sage supports the use of the BASIC[18] programming language to elucidate mathematics concepts in secondary school mathematics projects. Students are asked at the end of each chapter in the mathematics text to write programs for various concepts.[19]

The APL programming language (selected for use in this document) has been hailed for its concise notation and generalizability to arrays and for making it easier to appreciate patterns and structures in mathematics.[20] This convenient notation also greatly aids the process of analytic thought.[21] Iverson and Berry feel that this language is suitable for teaching topics in elementary algebra, coordinate geometry, statistics, calculus, sets and logic.

---

[18] BASIC is a programming language developed by John G. Kemeny (Dartmouth).

[19] Edwin Sage, "Problem-Solving with the Computer," (Newburyport, Mass.: Entelek, Inc., 1969), p. 244.

[20] Paul Berry, et al. "APL and Insight: The Use of Programs to Represent Concepts in Teaching," IBM Technical Report No. 320-3020 Philadelphia Scientific Center (March, 1973), 5-6.

[21] Kenneth F. Iverson, "APL As An Analytic Notation," IBM Philadelphia Scientific Center (paper presented at the APL V Conference, Toronto, Canada, May 15-18, 1973).

A sample of other persons advocating the use of APL to teach mathematics topics have been: Peter Caliganert who has written materials in Finite Group Theory, Paul Penfield who has proposed the use of APL in the study of derivatives in Calculus, and Michael Halpern who has used APL to look at permutations in Probability Theory.

Programming As A Conceptual Framework For Teaching Students

The previously mentioned educators felt that the aesthetic quality of mathematics as well as the utilitarian value could be realized more fully through the use of programming. But, there are also those educators who maintain that the mathematics or, for that matter, any discipline or technology should be subservient to the user. They feel that educational technology in general and computerized mathematics in particular have value only as they give students control over physical phenomena, or control over their own intellectual queries.

Among the educators who share this belief and hold the additional belief that programming and computers can play a central role in obtaining "student power" are Marvin Minsky and Seymour Papert, co-directors of M.I.T.'s Artificial Intelligence Laboratory.

Papert saw mathematics subordinate to the
acquisition of a "mathematical way of thinking," and both
computers and mathematics subordinate to the learner.
Papert wrote:

> Yes, one can use algebra as a vehicle
> for initiating students to the math-
> ematical way of thinking. But, to do
> so effectively, one should first identify
> as far as possible components of the
> general intellectual skills one is trying
> to teach; and when this is done it will
> appear that algebra (in any traditional
> sense) is not a particularly good vehicle . . .
> In our ideal mathematical laboratory the
> computer is used as a means to control
> physical process in order to achieve
> definite goals . . . [22]

Papert alluded to the "ideal" relationship between
student and programming technology when he presented his
grander vision of an educational system in which:

> . . . technology is used not in the form
> of machines for processing children but
> as something the child himself will learn
> to manipulate, to extend, to apply to
> projects, thereby gaining a greater and
> more articulate mastery of the world; a
> sense of the power or applied knowledge
> and a self-confidently realistic image
> of himself as an intellectual agent.[23]

Minsky shared his colleague's beliefs in the value
of computer programming in education. He went a step
further to optimistically predict that once the "powerful

---

[22] Seymour Papert, "Teaching Children to Be
Mathematicians versus Teaching Them About Mathematics,"
New Educational Technology (Spring, 1973), 9.

[23] Seymour Papert, "Teaching Children Thinking,"
New Educational Technology (Spring, 1973), 1.

concepts"[24] inherent in programming are elucidated and internalized by educators, American education may be radically changed. He predicted, "Eventaully, programming itself will become more important even than mathematics in early education."[25]

Thomas E. Kurtz and Larry L. Hatfield enthusiastically support the idea of turning computing power over to students. Kurtz' Dartmouth Secondary School Project uses the computer as the "perfect pupil" to help secondary and college level students clarify and organize their own thoughts. Hatfield's research focused on seventh grade students and used the same programming language as Kurtz (they both used BASIC). His results favored the treatment group using programming to clarify mathematical concepts.

The notion that the central role of the computer in instruction should be that of facilitator of concept clarification and that of tool of inquiry under students' direct control is a key concept in the studies of Berry and Bartoli. Both Berry and Bartoli adhered to "open use" of computers[26] and the "glass box" (as opposed to the "black

[24]Marvin Minsky, "Form and Content in Computer Science," New Educational Technology (Spring, 1973), 48.

[25]Ibid.

[26]Paul Berry, et al. "APL and Insight: A Strategy for Teaching," (paper presented to the Conference on APL, Institut De Recherche D'Informatique et D'Automatique, Paris, France, September 9-10, 1971), pp. 20-22.

box") computer programs.[27]   (See Chapter 3).

Howard A. Peelle of the University of Massachusetts strongly advocates the "glass box" approach to programming and adheres to the principle of student-controlled learning. He wrote, "In contrast to conventional computer assisted instruction (CAI) the glass box approach seeks to give the student control over his own learning."[28]

Finally, Luerhmann, Director of Project COMPUTe (Computer Oriented Materials Production for Undergraduate Teaching) and author of "Should the Computer Teach the Student or Vice Versa?" has this to say about the role of non-exploitative CAI and student control:

> . . . A student, programming a problem,
> and debugging it, is in a totally
> different mode of intellectual activity
> than another who is subject to lecturing
> or subject to a CAI lesson. When
> instructing the computer, a student
> is directing his own inquiry into the
> subject under study. He is the master,
> not merely the end user of some cost-
> effective new technology.[29]

---

[27]Ibid.

[28]Howard A. Peelle, "Teaching Children Thinking Viz APL," (paper presented at the APL V Conference in Toronto, Canada, May 15-18, 1973), pp. 0-1--0-2.

[29]Luehrmann, "Large Scale CAI--The NSF Program," 10.

Programming As A Conceptual Framework
For Teaching Teachers

"Bugs" and "debugging," "projects" and "subpro-

cedures," "iterative loops" and "recursion," "algorithms"

and "feedback," are expressions Papert and Minsky regarded

as <u>powerful ideas</u>.[30] Luehrmann referred to these as "new

intellectual structures."[31] These ideas are indeed

powerful and worthy of educators' considerations. Teacher

training programs and methods courses could probably profit

by using them as "epistemological primitives."[32] Kurtz'

Dartmouth Project lent credence to the use of computers and

computing heuristics in teacher education programs:

> Because we had to teach an ignorant
> machine, we were forced to break the
> process down into pieces, arrange these
> pieces in the proper order, and present
> them to our pupil machine and see if
> our instructions were presented in a
> logical, fool-proof way that it could
> follow. Before we made the effort to
> teach this pupil, we were forced to
> clearly understand the problem ourselves.[33]

---

[30]Papert, "Student and Computer--Toward a Deeper
Involvement," 16-17.

[31]Luehrmann, "Large Scale CAI--The NSF Program," 10.

[32]Papert, "Teaching Children to be Mathematicians,"
18.

[33]Thomas E. Kurtz, "The Computer as Pupil: The
Dartmouth Secondary School Project," NSF GW-2246, <u>Final
Report</u> (October, 1970), 18.

One known precedence exists to date for the use of programming in teacher training. Kenneth Iverson wrote a technical report titled "Introducing APL to Teachers" and gathered empirical data on the use of CAI_1 in teacher training in summer 1973 for high school and college level teachers.

For this document there exists no known previous research in the use of programming in elementary mathematics teacher training.

Conclusion

Technological strides made in the field of education represent, most often, a sincere effort to improve instruction and pupil understanding. What happens, though, to teachers and administrators when these new and exciting concepts are imposed on existing institutions? If no provisions are afoot to involve teachers (and students) in the whole technological thrust, these new developments would serve to only further "objectification, subservience, and isolation."[34]

Freeing up disciplines and students with technological catalysts may indeed mean abandoning the traditional

[34]Seymour B. Sarason, The Culture of the School and the Problem of Change (Boston: Allyn and Bacon, Inc., 1964), pp. 154-169.

conception of teaching, but, it would seem that teachers and students alike could, with the aid of computer technology, be allowed to go about intellectual pursuits as partners in knowledge rather than prisoners of it.

Using computers as "perfect pupils" as Kurtz suggests, may be just the gimmick needed to help teachers clarify their own thoughts. But, more importantly, it may place teachers in an unthreatening, success-oriented environment where self-confidence can be built, critical thinking can prosper, and "disadvantageousness" can be eliminated.

When research in the fields of computer-assisted instruction, mathematics education, and elementary teacher education was reviewed, a serious deficiency existed at the point of intersection of the three disciplines. Related literature in these areas pointed to the prospective "open-marriage"[35] of these partners. This author is about to begin the ceremony.

---

[35]Nena O'Neill and George O'Neill, Open Marriage, A New Life Style for Couples (New York: M. Evans and Company, Inc., 1972).

CHAPTER III

DESCRIPTION AND EVALUATION OF A PROGRAMMING COURSE

Introduction

In Chapter I, the need to revitalize mathematics teacher training programs was discussed. Possible improvements included: strengthening links between methods and content components, developing strategies to enrich curricula affectively, and increasing usages of computer technology and programming. Chapter II delineated previous attempts at introducing computer technology to teach mathematics, to teach students and finally, to teach teachers.

The major concern of this chapter is to describe an experimental programming course designed for prospective elementary school teachers. The course —— entitled "Mathematics and Programming" (MAP) ——was developed at the School of Education at the University of Massachusetts during 1972-73 in order to establish a stronger link between the mathematics methods courses taught at the School of Education and the mathematics content courses taught in the Department of Mathematics and Statistics. Specifically, this course was an attempt at developing effective

strategies for teaching prospective elementary school
teachers mathematics via A Programming Language.

Detailed in this chapter is an overview of the MAP
course including a description of the course participants
and the instructional support system, an outline of the
mathematics and programming content selected for the course,
and a discussion of the pedagogical strategies used to
develop the content materials. Also included in this
chapter is a report of a pilot programming course offered
in the Spring of 1973, an analysis of the findings of the
attitudinal and achievement instruments used to evaluate
the MAP course, and finally, a summary of findings and
personal reflections about teaching elementary school
teachers mathematics via A Programming Language.

Overview of MAP Course

The MAP course for prospective elementary school
teachers was offered under the auspices of the Teacher
Preparation Program Council (TPPC) at the School of
Education at the University of Massachusetts as part of the
Methods Potpourri component. The course was seven weeks in
duration, meeting twice a week for one hour and fifteen
minutes per session during September and October, 1973.
The course was 100 modules (1 University credit) and
carried a university lex number of 1525. Ordinarily, this
course was taken prior to student teaching.

MAP Course Participants

Students participating in the MAP course were
juniors and seniors with elementary education majors
preparing for elementary school certification. All
students participating in the MAP course were from small
New England towns. The ratio of females to males in this
course was nine to one.

None of the course participants had extensive
background in mathematics or programming. According to
a questionnaire (See Appendix A) administered on September
1', 1973, nine of the ten participants had taken the
Mathematics 110 (Elementary Techniques of Mathematics)
course offered in the Department of Mathematics and
Statistics. Of that nine, four had taken Mathematics 151
(Basic Concepts of Algebra) course also offered in the
Department of Mathematics and Statistics. None had taken
any course beyond the Mathematics 151 course.

Instructional Support System

Supporting this MAP course was an instructional
system which included computer equipment (hardware), a
system programming language (software), and instruction
materials (curriculum)—— See Chapter IV.

Hardware

The MAP course was made possible by the use of the Control Data Corporation 3600 time-sharing computer ("UMASS") housed at the University Computing Center, Amherst, Massachusetts. Datel tele-communications terminals (with APL type-balls) from Carterfone, Inc. were used as input-output devices——two terminals were located at the School of Education and some five additional terminals were available at the Graduate Research Center. Acoustic couplers from OMNITEC, Inc. and ordinary telephones from New England Telephone Company completed the on-line connections between the computer and the terminals.

Software

The central processing unit of the CDC 3600 time-sharing computer is monitored by a second, smaller computer--TEMPO, produced by GTE Information System. Several compilers and interpreters make up the translator for this machine. The interpreter on which this course depended was designed to process APL (A Programming Language) and was implemented in COMPASS on the CDC 3600 during 1970-71 under the direction of James Burrill, University Computing Center, University of Massachusetts.

Kenneth Iverson (1962) conceived the programming language, APL, used in this study. The notation used in this language is unambiguous, with simple and consistent

rules of syntax, and has provisions for powerful and direct use of arrays.[1]

Curriculum

The materials for instruction employed in this course consisted of a manual, and a supplemental guide. The manual, titled "APL for Teacher Learners," and the guide, titled "Supplemental Guide to APL for Teacher Learners," were written by this author. (See Chapter IV for detailed description of materials.

Mathematics Content Component

The mathematics content to be studied in the MAP course was selected from the "A" recommendations for elementary teacher training suggested by the National Council of Teachers of Mathematics (NCTM)[2] and from teacher's editions of current elementary mathematics textbook series. (See Chapter IV reference section in Supplemental Guide for complete listings of these series.) The following topics were included in the mathematics content coverage:

---

[1]See, for example, Kenneth E. Iverson, Algebra: An Algorithmic Treatment (Menlo Park: Addison-Wesley Publishing Co., 1971), p. 336.

[2]National Council of Teachers of Mathematics, Topics in Mathematics for Elementary School Teachers, Twenty-ninth yearbook, Washington NCTM, 1964. National Council of Teachers of Mathematics, More Topics in Mathematics for Elementary Teachers, Thirtieth yearbook, Washington NCTM, 1968.

I.   Set Theory
     Sets
     Membership
     Union
     Intersection
     Complement
     Cardinality of Sets

II. Number Theory
     Factors
     Composite Numbers
     Prime Numbers
     Least Common Multiples
     Greatest Common Divisors
     Relative Primes
     Fibonacci Sequences

III. The Decimal Numeration System
     Arithmetic Operations on Whole Numbers
     Arithmetic Operations on Integers
     Arithmetic Operations on Rational Numbers
     Arithmetic Operations on Real Numbers

IV. Structures of Other Numeration Systems
     Numbers in other Based Systems
     Numbers in other Modular Systems

V.   Geometry (Algebra of Geometry)

VI. Elementary Functions
     Arithmetic Functions
     Relational Functions
     Logical Functions
     Exponential Functions
     Miscellaneous Functions

VII. Problem Solving
     Iterative Notion of Procedures
     Iterative Notion of Subprocedures
     Heuristic for Debugging
     Recursive Procedures

Programming Content Component

The programming content studied in the MAP course was collected from materials written by Kenneth Iverson (1972), Sandra Pakin (1972) and Howard Peelle (1971, 1972).

The programming content included:

   I. Primitive Functions
       Monadic
       Dyadic

  II. Syntax
       Formulating Expressions
       Evaluating Expressions (R-L Rule)

 III. Defined Functions/Programs
       Niladic
       Monadic
       Dyadic
       Explicit Resultants

  IV. Arrays (Data Structures)
       Vectors
       Indexing
       Functions on Vectors
       Matrices

   V. Programming Techniques
       Branching
       Iteration
       Use of Subprocedures
       Debugging
       Recursion

Pedagogical Strategies

The specific objectives of the MAP course were:
(1) to provide prospective elementary school teachers with
sufficient programming skills to permit them to articulate
their thoughts about selected topics in mathematics to a
computer, (2) to suggest pedagogical strategies classroom
teachers could use when introducing computer programming
to elementary school children, and (3) to explicate
cognitive and affective behavioral objectives teachers
(facilitators) should stress when programming concepts are
employed in elementary school settings.

In order to achieve the last two objectives, it
was necessary to develop a meta-structure of pedagogical
strategies for introducing programming to the prospective
teachers themselves. This meta-structure was based on a
synthesis of educational philosophies. Believing that
"Teachers teach as they were taught"; borrowing from the
learning theories of Jean Piaget (1970) and Jerome Bruner
(1960); interweaving these theories with self-concept
philosophies discussed by William F. Johntz (1971), William
Purkey (1972), Rosenthal and Jacobson (1968); coupling these
philosophies with work in the field of affective development
advanced by Benjamin Bloom (1956, 1964), and Daniel Jordan
(1972); and finally drawing from the creative teaching
techniques devised by the Project S.E.E.D. (Special
Education for the Disadvantaged) staff at Berkeley and the
ANISA Project members at Amherst, Massachusetts, the meta-
structure representing an integration of the aforementioned
pedagogical strategies was developed. This structure was
believed to be potentially effective for teachers, because
it embodied those philosophies that have proven successful
with children.

Implicit in the course design were those learning
theories, self-concept principles and cognitive-affective
development schema mentioned above. Explicit in most
classes, however, were suggestions for creative teaching
practices the prospective teacher could utilize in future

elementary school classroom situations. These practices
came from checklists developed by ANISA and S.E.E.D staffs.

Excerpts from these two lists are as follows (See
Appendix D and E for complete checklists):

ANISA Checklist

1. Defer judgment
2. Emphasize process rather than product
3. Encourage speculation and guessing at
    appropriate times
4. Avoid being authoritarian
5. Introduce paradoxes to stimulate curiosities
6. Avoid making premature closures on activities
7. Encourage humor (laughter is a response to an
    unanticipated and therefore novel arrangement
    or integration of items or events . . .)[3]

S.E.E.D. Checklist

1. Do you start each of your classes with a
    conceptual review?
2. When you circulate around the room, do you
    mentally note which students, who normally
    do not participate in the verbal discussion,
    have correct answers?
3. Do you keep a log of each of your classes?
4. Have you created an atmosphere in which students
    are willing, even in very small number, to
    disagree with the majority view held by the
    class and/or teacher?
5. Have you ever indicated to your class that a
    really good question deserves even more credit
    and praise than a good answer?
6. Do you examine the children's technically incorrect
    answers to find out what they were thinking about?
7. If a child cannot answer a question, do you let
    him call upon another child to help him?[4]

---

[3]Jordan and Streets, <u>Releasing the Potentiality of
the Child</u>, p. 168.

[4]Special Elementary Education for the Disadvantaged
(S.E.E.D.) <u>Checklist</u>, Berkeley, California, 1971
(Mimeographed.)

The above checklists were distributed to course participants and they were asked to note deviations from these guideposts in the MAP course. At the end of the week, students were asked to discuss some of their observations.

Activities

The checklists discussed above were also used in the selection of activities for the MAP course. Only those activities that enabled intellectual development, enhanced self-concepts and encouraged creative curiosities were selected for use in the course. The selection of activities was also based on conformity to the basic philosophical underpinnings of this document——namely, students "learn by teaching" and "learning begets learning." (See Chapter I).

The course participants were encouraged to engage in open-ended, success-oriented, active learning and teaching activities. The following are examples of some of these activities: (See Chapter IV -- Supplemental Guide for more details on activities).

1. Role Playing - Acting in one act plays in sign-on and sign-off procedures.
2. Games (detailed in Appendix of Supplemental Guide in Chapter IV)
   "Knowledge-APL"
   "APL-500"
3. Self-Teaching manuals
   "APL for Teacher-Learners"
   "U-Programs and Mini U-Programs"
4. Computer Interactions
   Systematic experimentation
   Program production

5. Black Box Programs
    Mystery functions
    Guess my rule
6. "Glass-Box" Programs   (See Chapter IV)
    Hand Simulation
    Program modification
    Program examination
    Program production
7. Peer teaching, evaluation and discussion

Pilot Course

In the Spring of 1973, a pilot course was taught to
facilitate selection and development of curriculum materials
to be used in the Fall MAP course. This pilot course had
nineteen participants. It was six weeks in duration meeting
twice a week for one hour and fifteen minutes per session
in April and May, 1973.

The experimental design of this course allowed for
the necessary flexibility to try out several different
types of materials and teaching methods to determine
workable combinations of teaching strategies and content.

If students seemed to be unduly confused or
unchallenged by the materials, the materials were rejected.
If students found the activities exciting, challenging, and
provocative, the activities and materials were retained.
These and other subjective judgments were used to determine
the final collection of instructional materials.

At the completion of the pilot course, participants
who were junior and senior elementary education majors,

were asked by Spike Paranya, Coordinator of Methods
Potpourri, 1972-73, to evaluate the course by responding to
the following statements:

> Would you please evaluate the course or
> courses you have taken during the last
> six weeks. List what you have liked and
> found useful about each course, and what
> you feel can be improved, or have not
> liked in each course. Let us know if you
> feel the course should or should not be
> repeated.

The following are a few responses to the Potpourri

evaluation forms:

> A fun course. I enjoy it for now. If I ever
> get a chance to use it, I'll be thankful, but
> I don't expect to use it. How many schools
> have terminals?

> This was a very good introductory course . . .
> Good idea to introduce the computer step
> by step . . .

> The computer usage sections of the course was
> especially well presented. People who were
> afraid of computers were, within 3 weeks,
> simulating the insides of the computer for
> various math symbols (which is, of course,
> what children have to learn in an elementary
> math course).

> The computer course . . . very interesting, but
> I thought that the connection was not very
> definite how younger children could profit from
> learning math with a computer. It was a good
> experience just in the fact that I am now some-
> what familiar with a computer and just for
> general knowledge it was good.

> Using the computer forces a concretization (?)
> of the logic and thinking process involved. . .
> no short cuts due to past experience. A very
> useful aid to the potential math instructor.

In general, the evaluative responses revealed a

unanimous support for the continuation of the course.

Most students clearly indicated enjoyment of the course even though at times they were confused by some of the concepts presented. Constructive suggestions for improving the course design were to extend the course "possibly for one semester;" to show more "relevancy for elementary school students;" and to develop an "explanatory manual covering symbols used in APL."

Taking these suggestions and personal reflections about the course into consideration, curriculum materials were developed in the Summer of 1973 to be used in the Fall MAP course. (See Chapter IV for curriculum materials.)

Evaluation of MAP Course

An evaluation of the MAP course sought to determine the suitability and effectiveness of using A Programming Language to teach mathematics concepts to prospective elementary school teachers. A "course evaluation" questionnaire administered to course participants attempted to answer the suitability question; the question of effectiveness of the programming approach was assessed by "Attitude/Understanding Questionnaire," a "Cooperative Mathematics Test" (Form A and Form B) and an author-produced "Programming Articulation Test."

Course Evaluation Questionnaire

Participants were asked to give their reactions to the organization, content and the methodology employed in the MAP course by responding to a course evaluation questionnaire adapted from the University of Illinois course evaluation form (See Table I). This instrument contained 34 items in four major categories:

1. Course Content
   (Items 4,6,9,10,12,13,14,18,19,20,23,27,30,31,32)
2. Methods of Teaching
   (Items 3,5,8,10,11,16,23,25,29)
3. Organization of Course
   (Items 17,21,26,28,34)
4. Usefulness of Course
   (Items 1,2,7,15,22,33)

Students were asked to respond in four categories: strongly agree, agree, disagree, strongly disagree. Participants completing the questionnaire remained anonymous. It is hoped that anonymity produced a maximum likelihood of securing honest responses.

An analysis of the data showed that students thought very highly of the course content in terms of its appropriateness and its usefulness. All participants (one hundred percent) of the MAP course felt they had gained from taking the course, all felt that the content of the course was worthwhile, and all felt that the content of the course was excellent.

Reactions to the organization and methods of teaching the MAP course were more varied. The majority of the

TABLE   I

COURSE EVALUATION

	(Percentage who responded)			Mean Response	
	SA	A	D	SD	
1.	0	0	0	10	4	It was a waste of time.
2.	5	5	0	0	1.5	Overall, the course was good.
3.	2	8	0	0	1.8	More courses should be taught this way.
4.	4	6	0	0	1.6	The course held my interest.
5.	0	1	3	6	3.5	I would have preferred another method of teaching in this course.
6.	2	8	0	0	1.8	It was easy to remain attentive.
7.	0	0	4	6	3.6	Not much was gained by this course.
8.	5	5	0	0	1.5	The instructor encouraged the development of new viewpoints and appreciations.
9.	5	5	0	0	1.5	The course material seemed worthwhile.
10.	0	0	8	2	3.2	It was difficult to remain attentive.
11.	0	0	8	2	3.2	There was not enough student participation for this type of course.

TABLE I - Continued

	SA	A	D	SD	Mean Response	
12.	6	4	0	0	1.4	The content of the course was good.
13.	1	9	0	0	1.9	Held my attention throughout the course.
14.	0	0	4	6	3.6	Uninteresting course.
15.	4	6	0	0	1.6	It was a very worthwhile course.
16.	0	1	7	2	3.1	Some things were not explained very well.
17.	1	8	1	0	2.0	The way in which this course was taught results in better student learning.
18.	0	0	6	4	3.4	The course material was too difficult.
19.	0	0	2	8	3.8	One of my poorest courses.
20.	2	8	0	0	1.8	Material in the course was easy to follow.
21.	0	0	2	8	3.8	Course material was poorly organized.
22.	0	0	3	7	3.4	Course was not very helpful.
23.	2	8	0	0	1.8	It was quite interesting.
24.	4	6	0	0	1.6	I think that the course was taught quite well.

TABLE I - Continued

	SA	A	D	SD	Mean Response	
25.	0	0	5	5	3.5	I would prefer a different method of instruction.
26.	0	8	1	1	2.3	At times I was confused.
27.	2	8	0	0	1.8	Excellent course content.
28.	7	3	0	0	1.3	Generally, the course was well organized.
29.	0	3	6	1	2.8	Ideas and concepts were developed too rapidly.
30.	0	0	8	2	3.2	The content of the course was too elementary.
31.	0	1	9	0	2.9	Some days I was not very interested in this class.
32.	0	0	6	4	3.4	It was quite boring.
33.	1	9	0	0	1.9	The course was quite useful.
34.	3	7	0	0	1.7	I would take another course taught this way.

KEY

SA = STRONGLY AGREE
A = AGREE
D = DISAGREE
SD = STRONGLY DISAGREE

participants (ninety percent) preferred this method of teaching and all (one hundred percent) said they would take another course that was taught this way. Although there was agreement that the course was generally well organized, eighty percent said that at times they were confused by presentation of some of the materials. Informal interviews with participants revealed that part of this confusion was due to the accelerated pace taken in the later part of this course.

Attitude/Understanding Instrument

The attitude/understanding instrument was a questionnaire adapted from the various attitude differentials discussed in Charles Osgood's The Measurement of Meaning.[5] This instrument asked participants to circle monotonic increasing, monotonic decreasing, reverse curvatured, or linear functions to graphically illustrate their changes in attitudes and understandings occurring during the seven week MAP course. (See Appendix B for copy of actual instrument.)

The attitude/understanding questionnaire was administered at the end of the seven week course. Overall, students indicated that their attitudes toward computer programming changed significantly in a positive direction as a result of the MAP course. It was also revealed that

_____

[5]Charles E. Osgood, George J. Suci, Percy H. Tannebaum, The Measurement of Meaning (Urbana: University of Illinois Press, 1967).

understandings about mathematics, programming, computers

and teaching techniques showed positive increase.

Attitude/Understanding Questionnaire

Following is the frequency of responses to each item

of the attitude/understanding questionnaire. (See key below.)

1. How did your <u>attitude</u> toward <u>mathematics</u> change during
   this seven week course?

   RD - 0
   SD - 0
   ID - 0
   NC - 1
   DI - 3
   SI - 6
   RI - 0

2. How did your <u>attitude</u> toward <u>computers</u> change during
   this seven week course?

   RD - 0
   SD - 0
   ID - 0
   NC - 0
   DI - 0
   SI - 8
   RI - 2

3. How did your <u>attitude</u> toward <u>programming</u> change during
   this seven week course?

   RD - 0
   SD - 0
   ID - 0
   NC - 1
   DI - 1
   SI - 7
   RI - 1

4. How did your <u>attitude</u> toward the <u>teaching approaches used</u>
   in the "Programming Course" change during this seven
   week course, Namely:

a. What was your attitude toward manual "APL
   FOR TEACHER/LEARNERS"?

RD - 0
SD - 0
ID - 0
NC - 1
DI - 3
SI - 4
RI - 2

5. What was your attitude toward role-playing (simulated
   sign-on, sign-off, etc.)?

RD - 0
SD - 0
ID - 0
NC - 2
DI - 2
SI - 4
RI - 2

6. What was your attitude toward "Glass-Box Programs"?

RD - 0
SD - 0
ID - 0
NC - 3
DI - 3
SI - 3
RI - 1

7. What was your attitude toward Games (KNOWLEDGE-APL,
   APL/500 games)?

RD - 0
SD - 0
ID - 0
NC - 3
DI - 0
SI - 6
RI - 1

8. How did your mathematics understanding change during
   this seven week course?

RD - 0
SD - 0
ID - 0
NC - 1
DI - 3
SI - 5
RI - 1

9. How did your <u>programming understanding</u> change during
   this seven week course?

   RD - 0
   SD - 0
   ID - 0
   NC - 0
   DI - 0
   SI - 8
   RI - 2

10. How did your <u>understanding of teaching techniques</u>
    change during this seven week course?

    RD - 0
    SD - 0
    ID - 0
    NC - 2
    DI - 2
    SI - 6
    RI - 0

<u>KEY</u>

RD SD ID NC DI SI RI
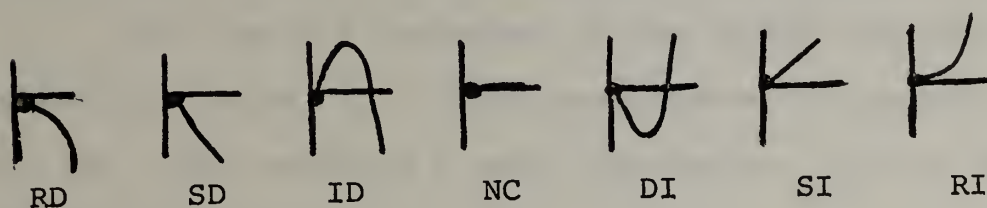
RD = Rapid Decrease
SD = Steady Decrease
ID = Increase-Decrease
NC = No Change
DI = Decrease-Increase
SI = Steady Increase
RI = Rapid Increase

An analysis of the results showed most of the responses falling into the "steady increase" and "rapid increase" categories, with "attitude toward computers" and "programming understanding" showing the greatest number, each having 10 responses clustered in these two categories. "Attitude toward programming" showed the next highest number of responses falling into the last two categories. Attitude change toward the use of "APL games" ranked a close third, with a total of seven responses, in the last two categories. "Attitude toward math," "the use of the APL manual," "the use of role-playing," "understanding math," and "understanding of teaching techniques" ranked roughly evenly in number of responses in the categories representing increases, although the distribution of responses varied.

No item had responses in the "rapid decrease," "steady decrease," or "increase-decrease" categories. All but two items received a small percentage (30%) of "no change" responses.

Mathematics Understanding Test

The instrument measuring mathematics understanding was a standardized Cooperative Mathematics Test titled "Structure of the Number System." This test was distributed by Educational Testing Service and was designed to test modern mathematics understandings.

Since the Cooperative Mathematics Test--"Structure of the Number System"--is a measure of developed abilities, its content validity was of primary importance. Upon examining several standardized tests of modern mathematics understanding, this test was found to be the most appropriate instrument available to test the mathematical content --namely, concepts in Number Theory--which most closely parallelled the substance of the programming problems chosen in the MAP course.

The reliability of this instrument was computed using the Kuder-Richardson Formula 20. The reported reliability of .82 for Forms A and B was more than adequate for these purposes.

A pretest of mathematics understanding was administered on September 11, 1973. Students were asked to complete the forty question multiple-choice Form A. A posttest was administered on October 23, 1973, with students being asked to respond to forty questions on

multiple-choice Form B. The results of the pre and post

tests for each individual student are as follows:

Pretest Scores	Posttest Scores
34	37
34	36
26	33
25	33
29	32
22	25
21	24
17	22
17	18
15	17

The range for pretest scores was 19, and the range

for posttest scores was 20 points. The mean of the pretest

was 24 and the mean on the posttest was 27.7. The standard

deviation for pretest scores was 6.50 and the standard

deviation for the posttest scores was 7.01. Using the

Pearson-Product Moment Correlation, there was a .95

correlation between the pretest and the posttest scores.

The sample size was not large enough for drawing

conclusions, but the following observations were suggested.

There was a 1-8 point increase on all students' pre and

posttest raw scores and an average point increase of four.

Compared to national school norms (based on samples of

seventh and eighth grade students enrolled in special

programs in modern mathematics) the average pretest score

of MAP participants fell in the eighty-eighth (88th) mid

percentile rank of seventh graders and seventy-third (73rd)

mid percentile rank of eighth graders. The average posttest

score of MAP participants fell in the ninety-fourth (94th) mid
percentile rank of seventh graders and eighty-fifth (85th)
mid percentile rank of eighth graders.

Programming Articulation Test

To assess MAP participants' achievements and
knowledge of programming, an author-produced test was
administered at the end of the seven week period (See
Appendix C). This test had twenty-two (22) items weighted
such that the total possible number of points was thirty-two
(32). Each item fell in one of the following categories:

1. Function Examination
   (Primitive Functions)   (items 1-16) 50% of test
   (Defined Functions)   (items 17,18,19) 18% of test

2. Function Modification   (item 20) 10% of test

3. Function Production   (items 21,22) 22% of test

Since sixty percent (60%) of the total class time
was spent in function examination and approximately twenty
percent (20%) of the time was spent in function modification,
with the remaining twenty percent (20%) spent in function
production, the percentage distribution above seemed to be
appropriate.

The results of this "Programming Articulation"
(PA) test was as follows:

Raw Scores: 19 18 12 20 15 10 13 12 10 5

The range of the PA test was fifteen (15). The mean
on this instrument was 13.4 and the standard deviation was
4.43.

Correlation Between "Mathematics Understanding" and
"Programming Articulation"

The programming articulation test scores of
participants were correlated with the mathematics under-
standing posttest scores using the Pearson-Product Moment
Correlation procedure. The correlation coefficient between
these two sets of data was .86, indicating that there was a
high positive relationship between individuals' under-
standings and programming articulation.

Personal Reflections

It is always rewarding to witness the timid unsure
student gain in self-assuredness to emerge competent and
confident learner. The beauty of this transformation lies
in the knowledge that the change process means a letting go
of securities and an embracing of momentary insecurities,
while leaving the self open to criticism. It is difficult,
if not impossible, to capture in writing or relay with any
degree of accuracy to non-eyewitnesses these instances of
vulnerabilities shared between learners and observers.
Although the findings reported on the aforementioned
instruments suggest positive kinds of changes in attitudes
and competencies, the statistical data only expresses
static states of a learner's development, while empirical
techniques come closer to capturing the instances of change.
This author witnessed two basic kinds of changes: changes
in cognitive growth and changes in affective growth.

Students in the MAP course and the pilot course
shared moments of their vulnerabilities with this author.
Most of the students came in the course expressing verbal
and nonverbal fears of computers and mathematics, but in
observations of these students throughout the course, some
of these fears were assuaged. Students who were literally
frightened of the terminal (and unseen computer) as well as
students hating mathematics began to steadily develop

positive feelings and attitudes about computers, programming
and mathematics. This suggests an important consequence
to utilizing computer technology in learning a "prestigious"
discipline like mathematics—— namely, dispelling myths
and fears about computers and mathematics by having people
actually come to realize that computer and mathematics
competency is not beyond the grasp of the layman by them-
selves becoming competent programmers. (Elitists in the
fields of mathematics and computer technology may have
self-serving reasons for continuing to preserve the
mystiques built up around computers and mathematics, but
we must be about the task of convincing them that this is
a dangerous posture.)

Another observable change in some of the students of
the MAP course and the pilot course was a gradual change
from "pawns" to "origins"[6] as Richard de Charms expressed
in "Origin Pawns and Educational Practice," or from
"producers" to "thinkers"[7] as John Holt wrote in How Children
Fail. The process of changing from something being
manipulated to the manipulator of things is a lonely,
frightening process, but some students during the

---

[6]Richard de Charms, "Origin Pawns and Educational
Practices," Washington University (Mimeographed.)

[7]John Holt, How Children Fail (New York: Pitman
Publishing Corporation, 1964).

"Function Definition" portion of the MAP and pilot course began to enjoy the challenge of defining things for themselves. Other students resisted this mode of creative-original thinking. Of these resistors, some expressed a mild dislike for the latter portion of the course and one student actually commented, "Writing programs makes you think too much."[8]

Finally, an informally held hypothesis--that changes occuring in mathematics understanding may be due to ability to articulate these understandings--was not proven nor disproven during the MAP or pilot course. Neither empirical nor statistical data supported the hypothesis, but statistical data did suggest that the converse of the statement may be true. This was evidenced by the high positive correlation between the "mathematics understanding" and "program articulation" scores. The fact that no conclusive evidence about this hypothesis was made available through the data analysis does not preclude the fact that the hypothesis is a provocative one and worthy of additional consideration. (See Chapter V, Summary, Conclusions, and Recommendations.)

---

[8]Some of the confusion reported on the "Course Evaluation" instrument may have been due in part to the rapidity of presentation of some materials, but some of the confusion was expressed when students were unwilling to try to think on their own. The safe standard kind of comment, and many of us are guilty of using it, was "I don't understand" or "I am confused."

C H A P T E R   I V

CURRICULUM MATERIAL FOR MAP COURSE

Described in this chapter are materials written by this author for use in the MAP course. (See Chapter III for description of course). These materials include a manual titled —— "APL for Teacher-Learners and a facilitator's guide titled—— "Supplemental Guide to APL for Teacher-Learners."

The "APL for Teacher-Learners" manual assumes that users have had no previous exposure to programming and that users are reasonably knowledgeable of number relations taught in the early elementary grades.

There are eight chapters in the "APL for Teacher-Learners" manual. Each chapter is building on the concepts from previous chapters——thus the skeleton appearing at the beginning of each chapter seems to be puting itself together. The first six chapters deal with symbols and syntax of primitive functions in APL. Chapter seven introduces defined functions in the language. And, chapter eight contains selected programs from elementary mathematics

texts series to be examined, modified and produced by
users.

The supplemental guide contains summaries of
chapters in the manual, cognitive and affective skills
facilitators should attempt to reinforce, suggested
activities and suggested ways of evaluating user's progress.

SUPPLEMENTAL

GUIDE

TO

APL

FOR
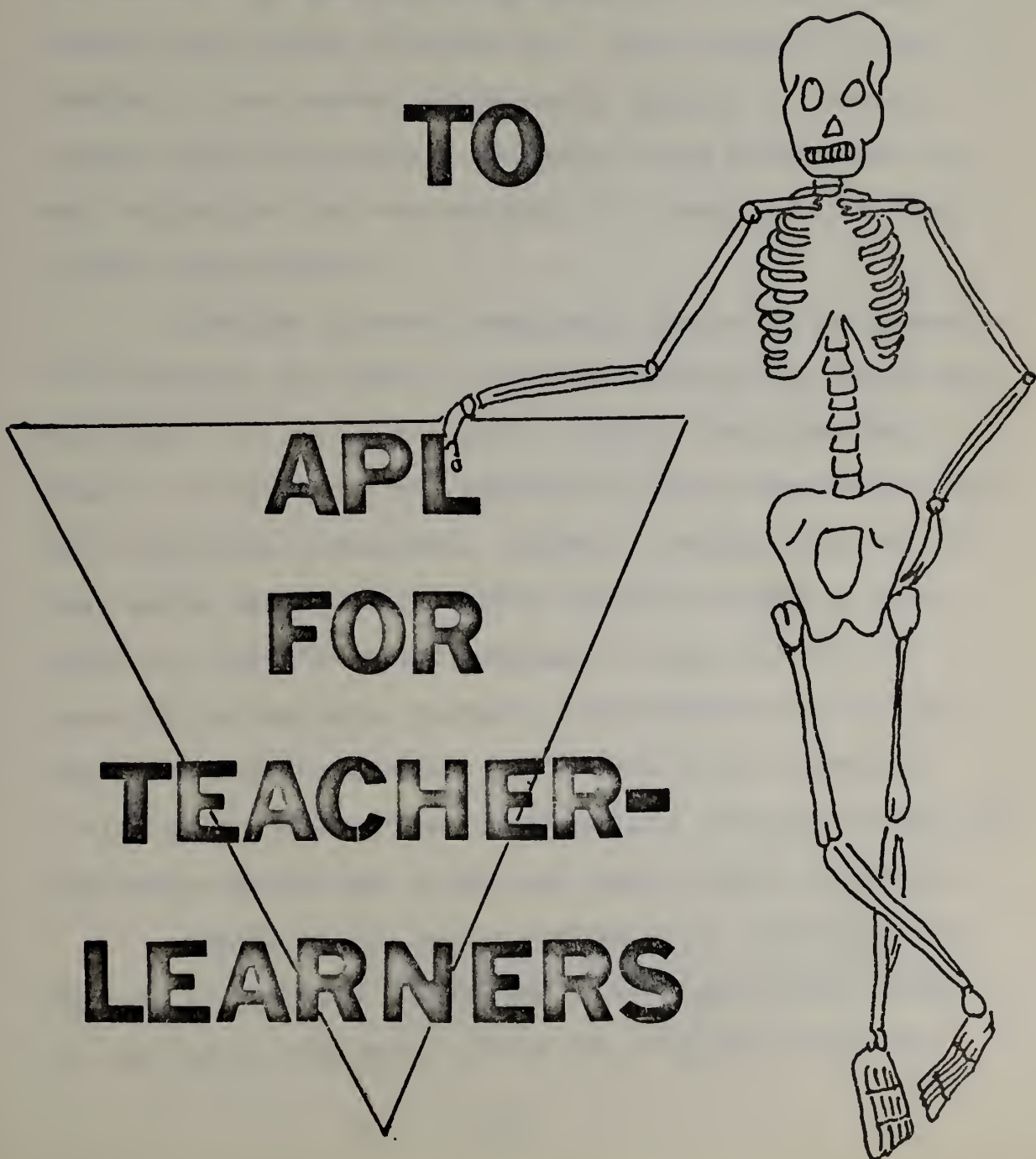
TEACHER-

LEARNERS

PREFACE

In the late 1950's and the early 1960's politicians and educators in this country felt an urgent need to change the emphasis in public school education to include more science and "modern mathematics." With national pride choking in the exhaust emissions of Sputnik I, national leaders began to strongly recommend "mass production" of more scientists and mathematicians to assure our victory in the "Space Race."

Textbook writers immediately jumped on the proverbial bandwagon and began to produce "Science for Today" and "New Math" series by the score. School administrators began to adopt these new series and thrust them on teachers and traditional curricula. Parents, feeling left out of this whole mushrooming process, began to throw up their hands and say, "Whatever happened to good old 2+2=4?" Doubtful as teachers, parents, and students were of the value of "Modern Science" and "Modern Math" offerings, "science proficiency" and "mathematics literacy" became the announced educational goals for public school education.

Coming on the heels of "New Math" was the introduction of computer technology to the educational scene. In the 1960's and early 1970's CAI (computer-assisted

instruction) projects began to spring up across the country with promises of cost-effective individualized instruction for the masses. Again, textbook writers, educators, local school boards, as well as federal policy-makers seemed "receptive to the plans of CAI proponents for using the technology of computing as a cost-effective delivery system for instruction in math or remedial English . . ."[1] But, few educational leaders realized that computing itself could be a valuable educational experience.

Mass computing literacy did not become the agreed-upon educational goal for the decade. Computer programming courses were restricted to vocational educational students at one end of the spectrum and to promising professional programmers at the other. Arthur W. Luehrmann of Dartmouth, asked, "If the computer is so powerful a resource that it can be programmed to simulate the instructional process, shouldn't we be teaching our students mastery of this powerful intellectual tool? Is is enough that a student be the subject of computer administered instruction--the end-user of a new technology? Or, should his education also include learning to use the computer? . . ."[2]

Many educators seemed to have turned a deaf ear on Luehrmann's questions, but we as teachers and parents

---

[1] Arthur W. Luehrmann, "Should the Computer Teach the Student, or Vice Versa?" (paper presented at the Spring Joint Computer Conference, 1972), p. 410.

[2] Ibid.

should not sit idly by and have educational policy-makers dictate whether computer literacy is necessary for our youths.

Questioning teachers, parents, and students have an obligation to find out for themselves if computer literacy will truly be needed to function in the world of tomorrow. And, they must ask, "Computer literacy for what end?" If persons attempting to answer these questions talk only of literacy for control of the environment or of aggressive nations--as people did when attempting to answer the question "Why mathematics literacy?"--we must remain leary of their motives. If persons answering these important questions acknowledge environmental and national defense concerns, but also include concerns for improvement of the individual--improved self-concepts and self-controls--we as guardians of the future can rest assured that our heirs will have the tools they need to control their own destiny and safeguard humanity.

P.C.E.

TABLE OF CONTENTS

PART I

INTRODUCTION

INTRODUCTION

"APL FOR TEACHER-LEARNERS" is a programming manual for elementary school students and their teachers. The manual introduces APL (A Programming Language) to novice programmers in a hierarchically sequenced self-teaching format. This manual is intended to provide users with programming skills sufficient to program selected topics from elementary school mathematics curricula.

An intellectual partnership between teachers and students is suggested by the phrase "teacher-learners" in the title of the manual. This phrase also suggests a departure from the commonly held view of teachers as omnipotent. Namely, it suggests that teachers are also learners.

The teachers using this manual will learn by teaching mathematics concepts to an "ignorant" computer. And, they will also learn how learners learn by putting themselves in the roles of students. Since teachers will be learning a new programming language and will have old concepts to re-master, they will be given "first-hand" experience at learning to become competent learners.

The student is not to be ignored in these programming pursuits because the student is the most important focus of these endeavors. Students will also be

2

"teacher-learners." They will be in the business of
learning how to learn and learning by teaching.

Turning the power of computer technology over to
"teacher-learners" who typically have been powerless in
authoritarian educational systems potentially can open up
new avenues of learning, as well as create new confidences
in selves.

The major objectives of this "Supplemental Guide
to APL for Teacher-Learners" are to:

1. Acquaint facilitators with the cognitive
    skills and processes needed to become
    literate programmers and with the affective
    skills needed to become competent teacher-
    learners;

2. To suggest activities facilitators can arrange
    to maximize teacher-learners' interactions
    with the most powerful tool humankind has
    ever known——THE COMPUTER.

WHO SHOULD USE THE MANUAL

The materials contained in "APL FOR TEACHER-LEARNERS" are geared towards upper elemetary school students and their teachers. Because of a spiraling[3] mathematics and programming curricula, the manual may also be used with junior high school students.
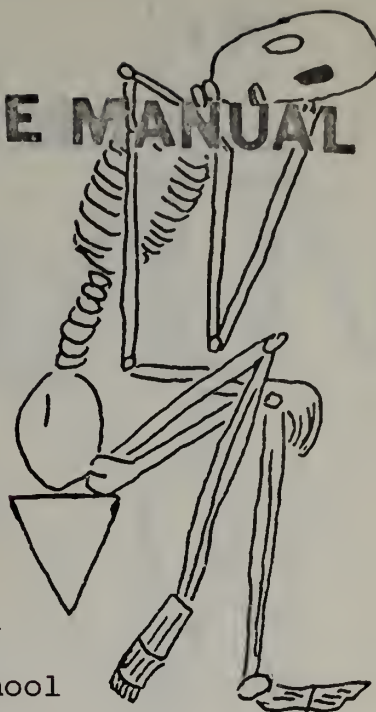
The "APL FOR TEACHER-LEARNERS" manual assumes that users have had no previous exposure to programming and that users are reasonably knowledgeable of number relations taught in the early elementary grades.

These materials were field tested on prospective elementary school majors at the University of Massachusetts at Amherst and on selected fifth and sixth graders at Mark's Meadow Elementary School in Amherst. The materials were found to be particularly well-suited for these two sample groups.

---

[3]Spiraling curricula refers to the practice of teaching concepts at each grade level with an increased amount of sophistication each year.

4

WAYS OF USING THE MANUAL

The "APL FOR TEACHER-LEARNERS" manual can be used in the following ways:

1. as a <u>textbook supplement</u> to elementary school mathematics series; (See reference section of Supplemental Guide.)

2. as a <u>workbook</u>; (There are places in the manual for students to write responses that they feel the computer will give.)

3. as a <u>reference book</u>; (Occasionally references to symbols or functions are made while programming at the terminal. An index is available at the end of the manual to facilitate these referals.)
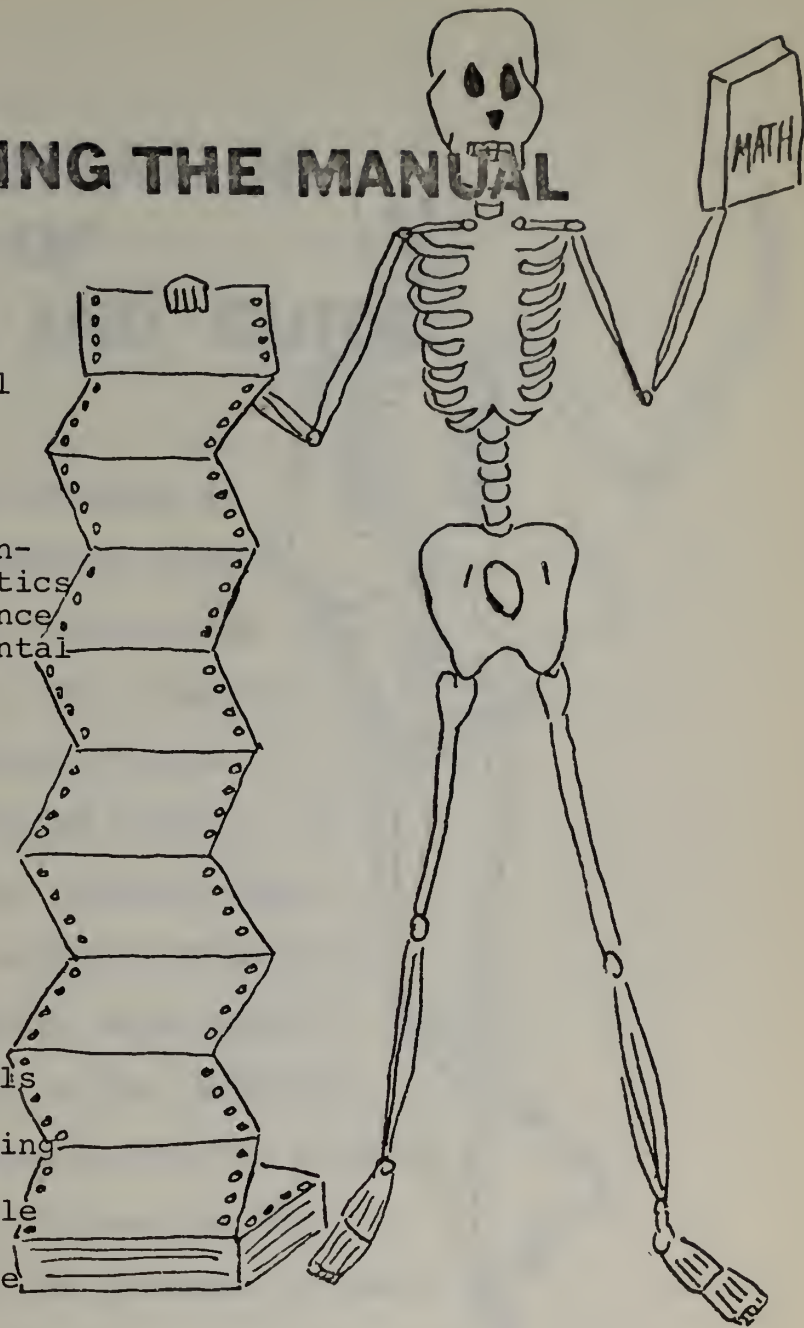
ORGANIZATION OF MANUAL AND GUIDE

THE MANUAL

There are eight chapters in the "APL FOR TEACHER-LEARNERS" manual. Each one builds on the concepts from previous chapters--thus, the skeleton appearing at the beginning of each chapter seems to be putting itself together. The first six chapters deal with the symbols and syntax of primitive functions in APL. Chapter seven introduces defined functions in the language. And, chapter eight contains selected programs from elementary school mathematics texts to be examined, modified, and produced by users.
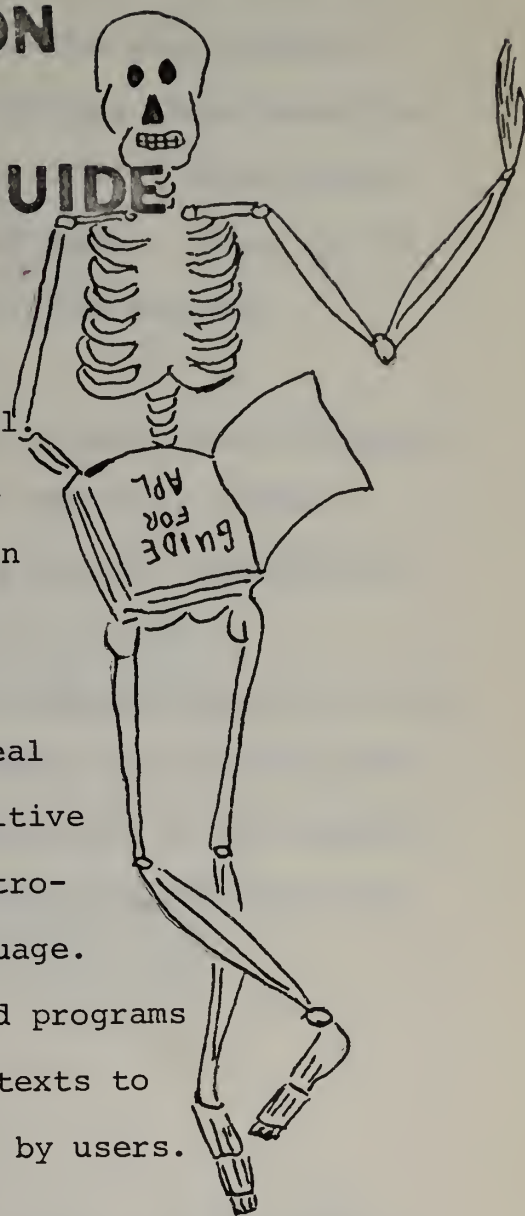
THE SUPPLEMENTAL GUIDE

This guide contains summaries of each chapter in the "APL FOR TEACHER-LEARNERS" manual. These summaries are intended to give concise statements about the content and aims of the chapters.

Following each summary is a list of new terms and symbols introduced in the chapter. Next, there will be a list of cognitive skills[4] the users should have acquired at the end of each chapter. Where possible these cognitive skills will be coupled with affective skills[5] facilitators should try to reinforce in teacher-learners. These skills include managing frustrations, coping with anxiety, expressing joy, and other feelings.

Activities will be suggested to give users maximum interactions with peers and with the computer. These activities will include games, role playing, "glassboxes," and programs devised by teachers.

Finally, ways of evaluating student progress will be listed. This evaluation will be student-oriented and will include suggestions for student evaluations of one another, class evaluations, informal comparative evaluations, and self-evaluations.

---

[4]Daniel Jordan and Donald Streets, of the ANISA Project at the University of Massachusetts developed the list of cognitive processes used in this text.

[5]The affective processes referred to in this guide were also developed by ANISA project staffers.

PART II

CHAPTER

OUTLINES

CHAPTER 1

GETTING THE COMPUTER'S ATTENTION
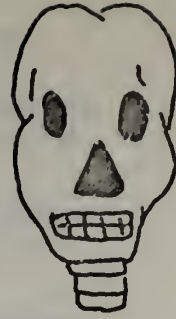
Chapter Summary

This chapter depicts the on-line connections between the computer and the terminal. Ordinary telephones (New England Telephone Company) are used with acoustic couplers (OMNITEC, Inc.) to establish communications with a CDC 3600 Time-Sharing Computer ("UMASS"). Datel tele-communications terminals with APL type-balls (Carterfone, Inc.) transmit signals via telephones to the computer. Similarly, the computer transmits responses to users via the same telephone-terminal hook-ups.

One act plays titled "Sign-On" and "Sign-Off" are included to help students learn signing on and off procedures.

New Terms and Symbols

On-line --   computer terminal hook-up via telephone
               and coupler which permits a user to sit
               at console, give instruction to the computer
               and get immediate reaction.

Sign-on --   establishing on-line communication
               with time-sharing computer.

Sign-off --   disconnecting on-line communication
               with computer.

9

User number -- distinct number (assigned by University Computing Center) which permits access to computer and by which user is registered and billed.

Code -- any four character combination of letters or numbers used to keep library confidential.

Library -- a place where user's workspaces are saved.

Workspace -- a block of space in the computer's storage area. The environment in which a user works.

Cognitive Objectives -- What We Should Know

Terminal Behaviors of Users:

1. Users should be able to sign on and sign off by themselves at the computer terminal.

2. Users should know the distinction between the actual computer and the terminal.

3. Users should be able to draw analogies between components of computer hook-ups and everyday experiences. For example:
   a. Terminal is to Programmer
      as
      Typewriter is to Secretary
   b. Telephone linkage is to Computer
      as
      Spinal cord is to Brain

Cognitive Processes for Facilitators to Reinforce:

1. Facilitators should reinforce the concept of analogy in relationship to computer technology.

Affective Objectives -- What We May Feel

Emotions and Feelings of Users:

1. Users may have an initial fear of the
   unknowns related to the terminal and
   the computer. (This is natural --
   especially for adults.)

2. Users may have an initial anxiety
   about not succeeding in the whole
   programming experience. (This may
   be healthy.)

3. Users may be elated about the prospects
   of prestige associated with learning
   to communicate with the computer. (This
   may be the result of novelty effect.)

Affective Processes for Facilitators to Reinforce:

1. Facilitators must help users assuage their
   fear by letting them:

   a. witness others at terminals
   b. look at the terminal and its
      keyboard and peripheral
      equipment until ready to try
      using it
   c. use pre-stored programs (e.g. games)
   d. free-play (e.g. try symbols)

2. Facilitators should help users manage their
   anxieties by keeping sign-on and sign-off
   instructions simple and clear and by providing
   many small success experiences with positive
   feedback. (Anxiety goes down when user can
   see for self that "it works" and "I can do it.")

Activities

1. Role playing, Sign-On, and Sign-Off procedures
   can be used to involve the entire class.
   (Let students volunteer to play the parts
   of the computer, the terminal, the telephone,
   and the coupler.)

11

2. Time should be alloted for:
   a. watching others;
   b. looking at the terminal and peripheral equipment;
   c. using pre-stored programs;
   d. playing freely.

3. An "Analogy Game" can be played with facilitators beginning by thinking of analogies between computer hook-ups and everyday experiences. (Try not to put people on the spot because this also produces unnecessary anxieties.)

Evaluation

1. Non-normative, self-evaluation techniques should be used at terminals.

2. Class evaluation of student's analogies in "Analogy Game" (Informal comments by peers.)

CHAPTER 2

INTRODUCING A PROGRAMMING LANGUAGE

Chapter Summary

This chapter introduces the user to the concepts of identifiers (variables), specifications, arithmetic operations, the catenation function, and value errors. A review of these concepts is found at the end of this chapter.
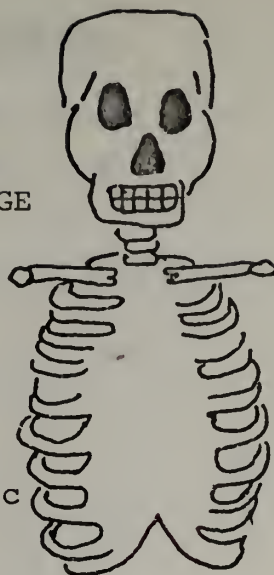
Highlights

1. To indicate when the user types, the computer automatically indents six spaces from the margin.

```
   'HELLO'   <————————————— User types
HELLO   <————————————————— Computer responds
```
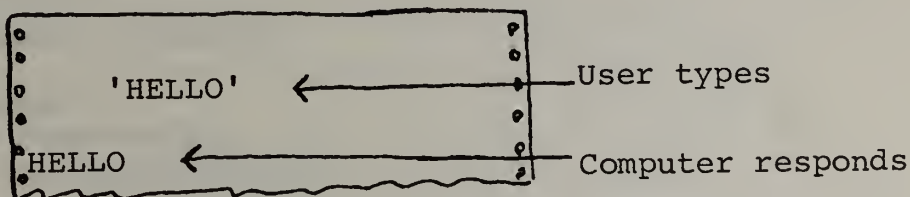
2. To get symbols appearing at the top of a key, hold shift key down while pressing key (like capitals on regular typewriter).

3. The return key must be pressed in order to enter an expression.

4. This symbol 🔵 in the manual asks user to respond as he thinks the computer will respond.

5. The user receives ERROR reports whenever the computer does not understand an expression. The VALUE ERROR report means you have not assigned a value to a particular variable.

13

New Terms and Symbols

Terms	Symbols
Literals	ABCD......Z
Numerals	01234.....9
Quotation Marks	' '

Specification
(the notation A ← 5
denotes that A is to
be made equal to 5.
The entire expression
is read "A is assigned
the value 5")

Symbol for Specification: ←

Arithmetic Functions   + - × ÷

Identifiers Z $\underline{H}$ MA $\underline{RATE}$ $F2$
(Identifiers used as
variables may be
single letters, under-
lined letters, combinations
of letters and numbers
[--all combinations are
permitted except those
beginning with S△ and
T△ and numbers.] No spacing is
used in identifiers.)

Catenation Function   ,

Variable
is a placeholder
for its replacements

Function
is a set of ordered
pairs (x,y) such that
for each value of x
there is at most one
value of y.

Cognitive Objectives -- What We Should Know

Terminal Behaviors of Users:

1. Users should know how to use quotation marks in APL expression.
2. Users should be able to assign values to variables.
3. Users should be able to interpret value errors when they occur.
4. Users should be able to link the values together using catenation function.

Cognitive Processes for Facilitators to Reinforce:

1. Facilitators should encourage inductive reasoning[6] processes because the manual is based on this type of reasoning. Users will be able to guess the responses the computer will give by looking at examples before and after their designated answer slot.
2. Facilitators should also encourage the interpolation[7] process (filling the gap) used in the format of the manual.
3. Facilitators should encourage the use of variables and the substitution process.

Affective Objectives -- What We May Feel

Emotions and Feelings of Users:

1. Users may feel isolated and lonely at the terminal—— or/may feel on the spot/being watched.
2. Users may have moments of frustration during this initial programming experience because of few successes. (Numerous error reports)

---

[6]Inductive reasoning involves deriving generalizations from an enumeration of collected data.

[7]Interpolation refers to inferences derived from context.

CHAPTER 3

PRIMITIVE FUNCTIONS--DYADIC

Chapter Summary

This chapter defines, gives examples and syntax structure of dyadic primitive functions. (A dyadic function is an APL function with arguments on both sides; i.e. 7 ÷ 3; 8 × 6; 4 - 5 6 ÷ 7 ; or 1 3 = 5 6.

This chapter also calls for systematic experimentation of selected APL dyadic primitives.

A cumulative review appears at the end of this chapter.

Highlights

1. Functions are performed on lists and with single arguments on each side.

2. Fractions are represented in decimal notation. 1/2 is expressed as .5, and 1/3 is expressed as .3333333333 (depending on the DIGITS command previously given) (See highlights in Chapter 4)

3. Relational functions <, ≥, ≠, etc. have a range (or give results) of zeroes and ones.

4. Logical functions have domain and range of zeroes (false) and ones (true).

5. Other dyadic functions include   * ⌈ ⌊ |

6. The minus sign and the negative signs are distinguished by their position on the typing line. The expression '3 - 8' uses the minus function while the expression '¯5' uses the negative symbol.
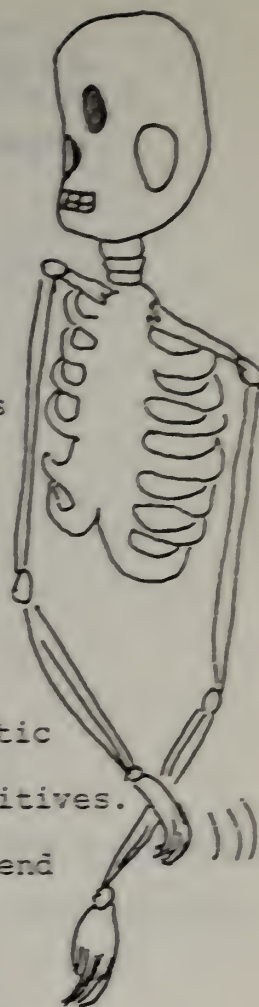
7. Domain errors indicate that a function is not defined for given values of arguments. (The expression $7 \div 0$ yields a domain error.)

8. Length errors occur when vector arguments are of unequal lengths.

New Terms and Symbols

Terms	Symbols
Relational Functions	< ≤ = ≥ > ≠
Logical Functions	∨ ∧ ~
Exponentiation	*
Residue Function	|
Minimum Function	L
Maximum Function	Γ

Argument -- any value assigned to a variable.

Domain -- the set of all possible arguments for a function.

Range -- the set of all possible resultants of a function.

Cognitive Objectives -- What We Should Know

Terminal Behaviors of Users:

1. Users should be able to form syntactically well-formed dyadic primitive expressions in APL.

2. Users should be able to predict with accuracy the computer's response to all the relational and logical expressions.

3. Users should be able to approximate values of resultants obtained from all arithmetic and exponential expressions.

4. Users should be able to determine what the minimum L, maximum Γ, and the residue | functions do by systematic experimentation.

5. Users should be able to hypothesize or make predictions about nature of unfamiliar functions.

Activities

1. "Guess My Rule"——Students will think
   up functions, give a few results from
   arguments and have the rest of the class
   try to guess their rules.

2. "Mystery Function"——Students will have
   to guess mystery functions pre-programmed
   by facilitator. (Black-box approach to
   programming.)

Evaluation

1. Class evaluation of users' rules in
   "Guess My Rule" game.

2. Self-evaluation in "Mystery Function" game.

CHAPTER 4

PRIMITIVE FUNCTIONS--MONADIC

Chapter Summary

    Selected APL primitive monadic
functions will be explored in this
chapter. Monadic functions are functions
with one argument, i.e. $\div 7$, $\times 7.2$, etc.

    The syntax for monadic functions is
given at the beginning of the chapter and users are asked
to systematically explore various monadic primitives to
determine their natures. Finally, a cumulative review
appears at the end of this chapter.

Highlights

    1. Monadic functions include: $+$ $-$ $\times$ $\div$ $|$ ι $\lfloor$ $\lceil$ $?$

    2. Two system commands which effect functions
       are as follows:

        a. The systems command)DIGITS determines
          the number of significant digits that
          will follow the decimal point in a decimal
          fraction.

)DIGITS   4

               $\div 3$

           .3333

)DIGITS   10 command yields

           .3333333333 for the expression $\div 3$.

        b. The systems command)ORIGIN determines
          whether one or zero will begin a list
          determined by the iota function:

CHAPTER 5

EVALUATING EXPRESSIONS --
RIGHT-TO-LEFT
EXECUTION
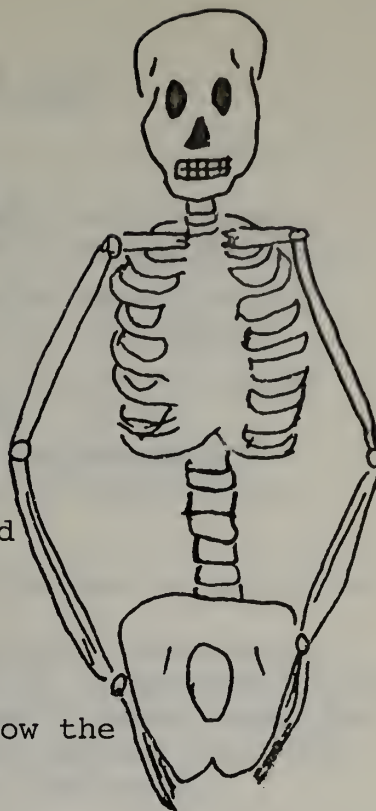
Chapter Summary

Now that students have learned
the rules for forming syntactically
correct expressions with primitive
functions in APL, it is time to see how the
computer goes about evaluating these
expressions. This chapter is devoted to the study of the
rules of evaluating expressions.

The general rule for evaluation is that every
function operates on the entire expression to its right.
Another rule is that parentheses take priority in any
expression. Inside parentheses must be evaluated first.

Highlights

1. Syntax errors occur when there are unequal
   numbers of right and left parentheses.

New Terms and Symbols

Terms	Symbols
Parentheses	( )
Right-to-Left Execution	

Cognitive Objectives -- What We Should Know

Terminal Behaviors of Users:

1. Users should be able to evaluate syntactically well-formed APL expressions with no parentheses.

2. Users should be able to evaluate syntactically well-formed APL expressions with parentheses.

3. Users should be able to identify syntactically ill-formed expressions in APL.

4. Users should be able to write expressions in APL using the least number of parentheses necessary for desired results.

Cognitive Processes for Facilitators to Reinforce:

1. Classifying techniques (identifying attributes through a process of abstraction or differentiation and grouping attributes to form classes) must be employed by users to determine whether functions are being used monadically or dyadically.

Affective Objectives -- How We May Feel

Emotions and Feelings of Users:

1. Some users may be very successful with the rules in this chapter and express positive feelings because of the brevity/simplicity of the rules. (They do not have to memorize a hierarchy of functions in order to evaluate an expression.)

2. Other users may become frustrated by these rules. This frustration may be due to user's inability to reject traditional ways of evaluating arithmetic expressions.

Affective Processes for Facilitators to Reinforce:

1. Continued emphasis on management of frustrations is necessary. Facilitators must provide users with numerous short, clear, and easy expressions to evaluate before giving them longer, more difficult expressions.

2. Those users who have successes with materials should be encouraged to help students having difficulties.

<u>Activities</u>

1. "APL/500" game -- (See Appendix of Supplemental
Guide)

2. "Knowledge-APL"-- (See Appendix of Supplemental
Guide)

<u>Evaluation</u>

1. Student evaluation of each other in both
games. (The computer should be the <u>arbiter</u>
in any disputes arising from playing the
games.)
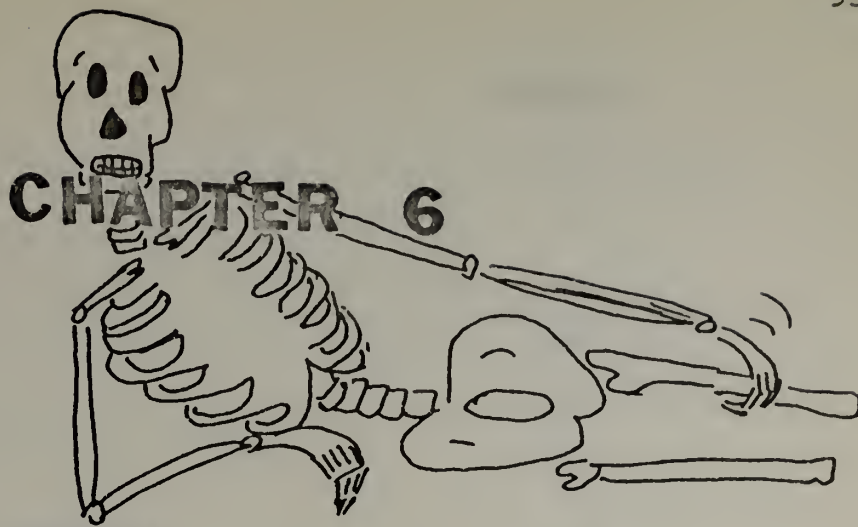
CHAPTER 6

FUNCTIONS OVER LISTS--VECTORS

Chapter Summary

This chapter explores selected dyadic and monadic functions on lists and explores indexing on lists. Users are encouraged to continue systematic experimentation with these new functions. A cumulative review is to be written by the users to find out "What Have You Learned?"

Highlights

1. Compression and Expansion functions need logical vector left arguments, i.e. $0\ 1\ 0\ /'APL'$

2. The reversal function ϕ is formed by typing the circle above the 0 on the keyboard, backspace and the verticle line above the M.

3. Index errors occur when there is a request for a nonexistent element of an array (vectors or matrices).

New Terms and Symbols

Terms	Symbols
Reductions	$+/\ \ -/\ \ \times/\ \ \div/\ \ \lceil/\ \ </\ \ \wedge/$

27

Terms	Symbols
Compression	/
Expansion	\
Membership	∈
Drop	↓
Take	↑
Shape (rho)	ρ
Brackets (for indexing)	[ ]

Cognitive Objectives -- What We Should Know

Terminal Behaviors of Users:

1. Users should be able to evaluate expressions using functions over lists.
2. Users should be able to determine what the reduction +/ , drop ↓ , take ↑ , and reversal φ functions do by systematic experimentation.
3. Users should be able to determine whether index errors will arise from particular expressions.
4. Users should be able to enumerate the functions they have learned in this chapter by making up their own Chapter Review—— "What Have You Learned?"

Cognitive Processes for Facilitators to Reinforce:

(See Chapter 3 of Supplemental Guide)

Affective Objectives -- How We May Feel

Emotions and Feelings of Users:

(See Chapter 3 of Supplemental Guide)

Affective Processes for Facilitators to Reinforce:

(See Chapter 3 of Supplemental Guide)

29

Activities

1. Make up Chapter Review——Users should make up
    a review of the chapter and give it to a
    classmate to try to work through.

2. "Guess my Word"——By employing indexing
    techniques users can have other students try
    to guess mystery words they have written.

3. Users should try to make up anagrams using
    indexing techniques.

Evaluation

1. Students should evaluate other student's
    Chapter Reviews after working through them.

CHAPTER 7

DEFINED FUNCTIONS

Chapter Summary

This chapter includes the following four sections: The Anatomy of Defined Functions, The Syntax of Defined Functions, Branching in Defined Functions, and Editing Defined Functions. Users are asked to examine and modify selected programs presented in this chapter. The review found at the end of this chapter asks users to pick out all the errors in a program.

Highlights

1. The symbol ∇ is used to start and end all defined functions. This symbol places you in definition mode. Up until now all work has been done in execution mode.

2. Quote-Quad ⍞ is used for literal input only.

3. Quad ⎕ is used for numeric or literal input.

4. The system command which allows you to delete a program from your workspace is)ERASE NAME.
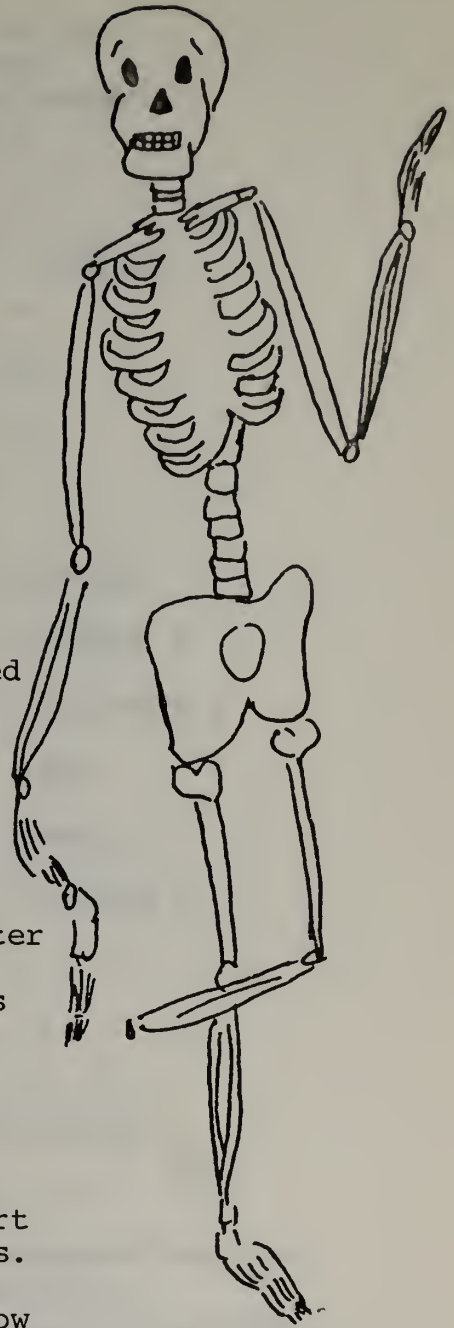
5. Endless loops may be gotten out of
   by pressing the ATTN (attention)
   button at the right of the keyboard.

New Terms and Symbols

Terms	Symbols
Del	∇
Quote-Quad	▯
Quad	□

Syntax of Defined Functions

Niladic Explicit Results	∇Z←NAME
Monadic Explicit Results	∇Z←NAME A
Dyadic Explicit Results	∇Z←B NAME A
Niladic no Explicit Results	∇NAME
Monadic no Explicit Results	∇NAME A
Dyadic no Explicit Results	∇B NAME A
Unconditional Branching	→
Conditioned Branching (format)	→( )/

Cognitive Objectives -- What We Should Know

Terminal Behavior of Users:

1. Users should be able to distinguish between
   the six types of defined functions.
2. Users should be able to distinguish between
   a conditional and unconditional branch statement.
3. Users should be able to "hand execute" selected
   defined functions in the Chapter. (Examine
   Programs)
4. Users should be able to modify selected
   defined functions in this chapter.

Cognitive Processes for Facilitators to Reinforce:

1. Facilitators should help users in their classification or categorization of defined functions and kinds of branching.

2. Deductive reasoning skills must be stressed by facilitators in "hand execution" of defined functions.

Affective Objectives -- How We May Feel

Emotions and Feelings of Users:

1. Users may be anxious again in these initial stages of defining functions (programs). [During function definition, execution of program is postponed so feedback (gratification or disappointment) is delayed. This waiting is a source of anxiety.]

2. Users' anxiety may give way to feelings of frustrations caused by unsuccessful work with defined functions.

3. Users may express a mild dislike for this new phase of learning to programming.[9] This phase calls for more individual work so the dislike may be due to (1) having to think for themselves or (2) having to be on their own.

4. Because work begins to become more individualized, users may again feel isolated and lonely in this phase of programming.

5. Users may become disappointed when their programs fail to run as expected.

Affective Processes for Facilitators to Reinforce:

1. Facilitators must help users manage their frustrations and anxieties by making sure that programming tasks are not beyond the scope of their programming abilities. (Not too great a disequilibrium).

---

[9]This is similar to the "math-hate" syndrome in some children.

2. Facilitators must try to inhibit the feelings of dislike for programming by again making sure that the programming tasks are not beyond student's abilities and by assuring many success experiences in these initial stages of defining functions (writing programs). Facilitators may also want to design special programming tasks for individuals to become "stars."[10]

3. Some users will not know where to start in this initial program definition stage so facilitators will have to encourage students to "guess" (take a stab at defining functions) and learn from own errors. These concrete consequences are better than making no attempt.

4. Feelings of isolation and loneliness must be coped with by users. Facilitators can help by keeping the terminal near classroom activities and by letting children have partners to help them examine their work.

5. Some users may have guilt feelings about unsuccessful attempts at modifying programs. They sometimes feel that they have "messed good (working) programs up." Facilitators must help students turn these feelings of guilt and failure into positive advantages. This can be done by having students look at actual results and hunt down errors.

Activities

CAUTION: Facilitators may need to reduce pace (similar to phase of first learning primitives). LEARNING TO DEFINE FUNCTIONS IS A MAJOR HURDLE.

1. A great deal of program modification must be done at this time.

2. Users should play the Advanced "Knowledge-APL" game. The rules in the original game are changed to allow students to define functions

---

[10] This "starring" technique has been used successfully by members of Project S.E.E.D. (Special Elementary Education for the Disadvantaged) under the direction of William Johntz.

to add to their "deeds to knowledge"
every time they pass go. The number
of lines in the program must be equal
to the number of times the player has
passed Go. The program must run before
it can be added to "deeds to knowledge."

3. Working directly at the terminal is important.
Let most of the programming activities
center around the terminals.

4. Group activities (not much work alone)
e.g. partnership.

5. Facilitators should place programs on the
bulletin board and encourage informal
evaluation of student's work.

Evaluation

1. Class scrutiny and hand executing of
modified functions should be encouraged.
(Evaluators should learn to offer
constructive criticism.)

2. Individuals should choose partners to
critique their work.

3. Facilitators should place programs on
the bulletin board and encourage
informal evaluations of students' work.

CHAPTER 8

PROGRAM PRODUCTION

Chapter Summary

In this chapter users are given many opportunities to create their own programs to do various mathematical tasks. Some emphasis will be placed on program examination and modification, but major emphasis will be on individual program production. Selected topics in mathematics are suggested by these materials, but users are encouraged to "dream up" their own problems to solve via APL programming.

Highlights

Major topics in mathematics include:

1. Set Theory
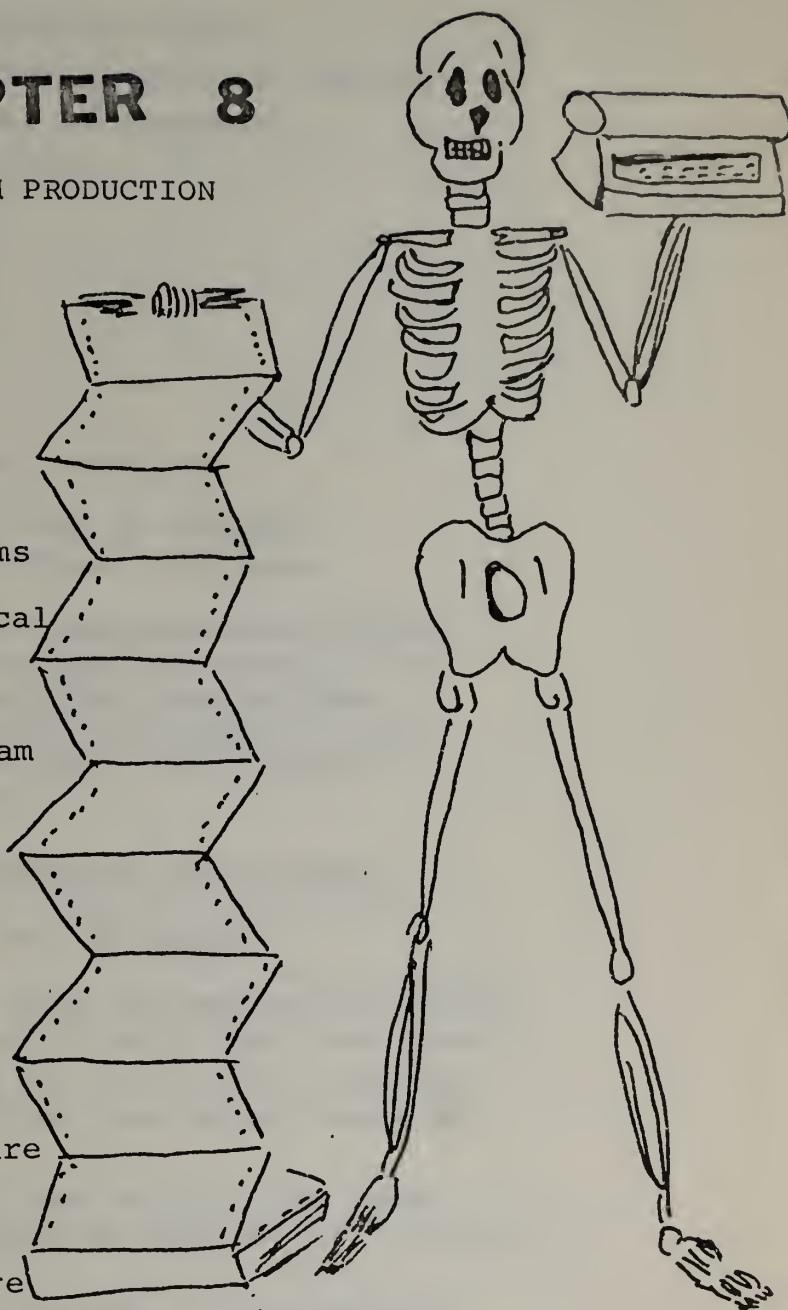
2. Number Theory

3. The Decimal Numeration System

4. Structures of Other Numeration Systems

5. Geometry (Algebra of Geometry)

6. Elementary Functions
     - arithmetic
     - relational
     - logic
     - exponential

New Terms and Symbols

Bugs -- Errors in defined functions

Debugging -- techniques used to correct
             errors in defined functions

Subprocedures -- defined functions used inside
                 other defined functions. (To
                 use functions inside other
                 functions, the syntax of the
                 program must yield explicit
                 results.)

Cognitive Objectives -- What We Should Know

        Terminal Behaviors of Users:

1. Users should be able to examine and modify
    existing programs to make them more general.

1. Users should be able to produce original
    programs to perform particular tasks in
    mathematics.

3. Users should be able to edit and debug
    their own programs as well as the programs
    of others.

4. Students should be able to use defined
    functions inside other defined functions.
    ("Sub-procedures")

Cognitive Processes for Facilitators to Reinforce:

1. Abstract reasoning will be required to help users create their own original programs.

2. Facilitators must encourage the development of deductive reasoning when users attempt to define their own functions.

3. Facilitators must encourage experimentations because flaws in arguments can be detected easily when error reports come from the computer.

4. Generalizing programs so that they include all possible cases in an argument is an important process for facilitators to reinforce.

Affective Objectives -- What We May Feel

Emotions and Feelings of Users:

1. Users may become frustrated by unsuccessful attempts at writing their own programs.

2. Waiting for feedback (gratification or disappointments) once programs are executed is an anxiety-producing experience for most users.

3. Some users may turn initial feelings of anxiety into dislikes or hostilities toward computers, people, or programming.

4. Some users may turn anxieties into positive feelings about computers, people, and/or programming.

Affective Processes for Facilitators to Reinforce:

1. Facilitators must help users manage their frustrations and cope with their disappointments by constantly reinforcing the important fact that errors in programs are simple bugs in thinking and not reflections of personal inadequacies. Facilitators must also be on hand (continuously circulating around students) to spot and avert some difficulties. (There is a delicate balance between helping, doing, and interfering.)

2. Since programming is an anxiety-producing activity, facilitators must check these anxieties and assist users in development of attitudes about programming which enable them to "move toward people" and computers "rather than away from them or against them."[11] For facilitators to check anxieties they must be consistent and constructive in feedback. [Inherent in most computer software (computer languages) are built-in checks on anxieties produced by programming. Error messages are consistently reported and in APL language the line and spot where the error occurs is also given.]

Activities

1. Much of the time should be spent writing, debugging, editing and revising programs.

2. Users should be able to go to "debugging clinics" (other users or facilitators) to get immediate help to identify bugs in programs.

Evaluations

1. Self-evaluation techniques should be used. Computer's consistent feedback will be best indication of own understandings.

2. Student evaluation should be encouraged.

3. Informal comparative evaluation techniques should be used.

---

[11] Daniel Jordan and Donald Streets, *Releasing the Potentialities of the Child*, p. 133.

REFLECTIONS

The success of these
materials depends on a facilitator
who bears in mind that
programming is a means of
communicating ideas as well as
feelings and that programmers may,
at times, be inarticulate in these
expressions. It will be on these
occasions of inarticulateness that
facilitators will have to convey
to students that errors (or bugs) in
idea-feeling expressions (programs)
are things to be "tracked down,
conquered, or tamed," but not things
to be internalized as reflectors of
personal inadequacies. The facilitators who maintain these
points of view will turn out "teacher-learners" who "are
not afraid to gamble, who are not afraid to experiment, who
are not afraid of the difficult and unknown." This will be
the greatest legacy we, as facilitators, can leave to the
"heirs of humanity."

39
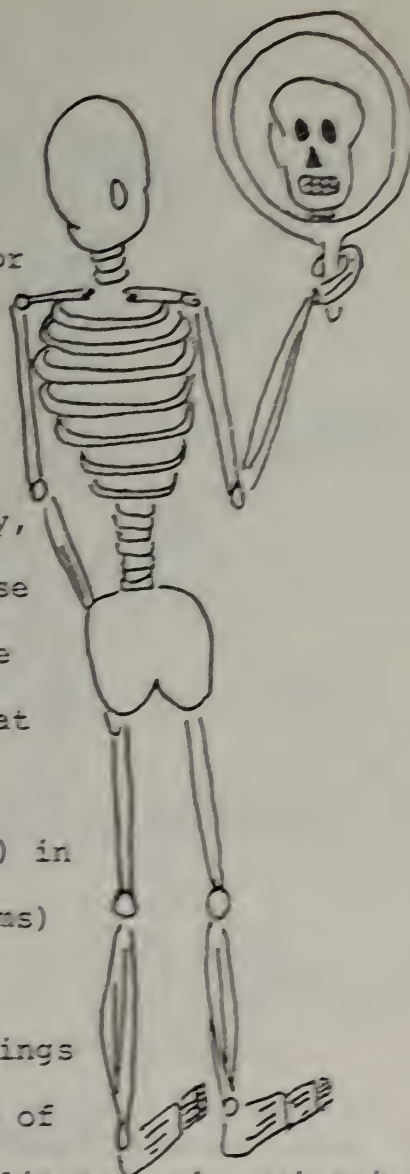
REFERENCES

Three APL texts were used in the selection of programming content materials in Chapters 1 - 7. Eight elementary textbooks series were used in the selection of mathematics content materials to be programmed in chapter 8. These references include:

APL References

Iverson, Kenneth E. "Introducing APL to Teachers." IBM Technical Report No. 320-3014, Philadelphia Scientific Center (July, 1972).

Pakin, Sandra. APL \360 Reference Manual. 2nd edition Chicago, Illinois: Science Research Associates, Inc. 1972.

Peelle, Howard A. "U-Programs." and "Mini U-Programs." Poughkeepsie, New York: Shared Educational Computer System, Inc., (SECOS), 1971, 1972.
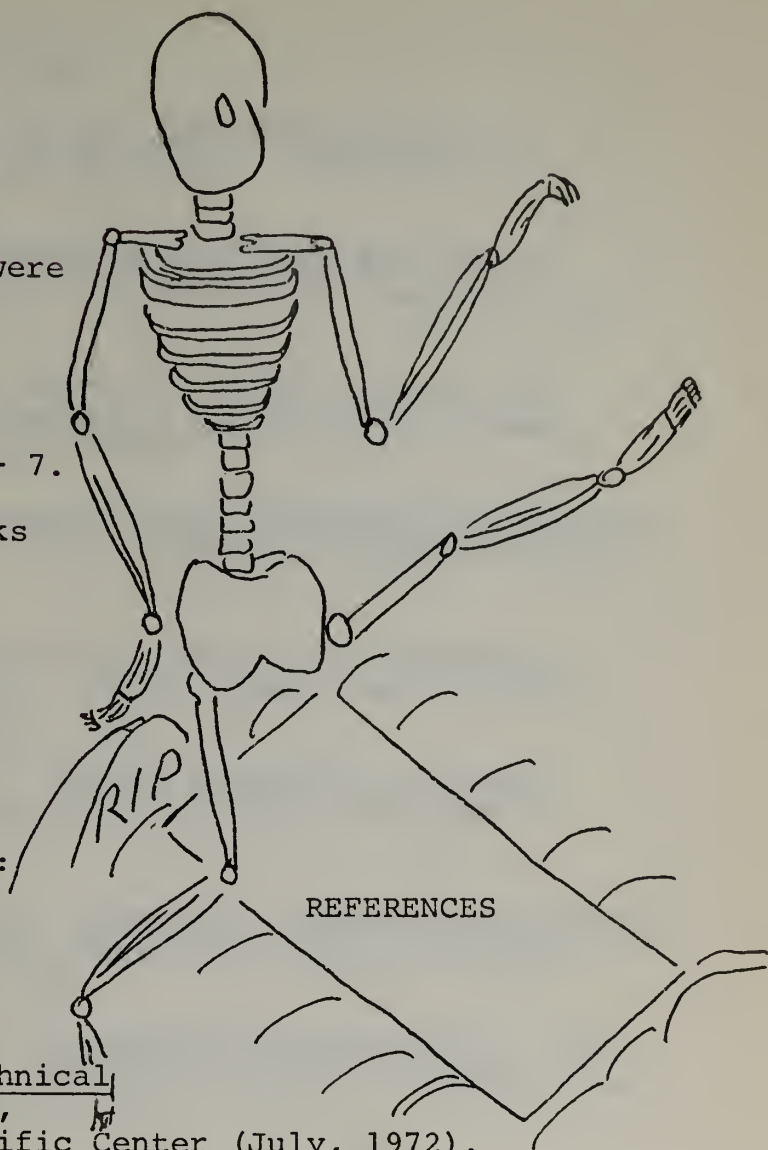
REFERENCES

Elementary Textbook References

D'Augustine, Charles H., et al. New Dimensions in
    Mathematics, Teacher's Edition 5. New York, N.Y.:
    Harper Row, 1970.

Deans, Edwina, et al. Teacher's Annotated Edition of
    Learning Mathematics. New York, N.Y.: American Book
    Company, 1968.

Dilley, Clyde A., Rucker, Walter E., and Jackson, Ann.
    Heath Elementary Mathematics 5. Lexington, Mass.:
    D.C. Heath and Company, 1972

Duncan, Ernest, et al. Modern School Mathematics Structure
    and Use 5. Boston, Mass.: Houghton Mifflin Company,
    1967.

Eicholz, Robert E., and Odaffer, Phares G. Elementary
    School Mathematics 5. 2nd ed. Menlo Park, Calif.:
    Addison-Wesley, 1968.

Morton, Robert L., et al. Modern Mathematics through
    Discovery 5. Morriston, N.J.: Silver Burdett
    Company, 1970.

Nichols, Eugene D., et al. Elementary Mathematics Patterns
    and Structures 5. New York, N.Y.: Holt, Rinehart,
    and Winston, Inc., 1966.

Payne, Joseph N., et al. Elementary Mathematics 5. 2nd
    ed. New York, N.Y.: Harcourt, Bruce, and World, Inc.,
    1968.

APPENDIX

Rules for Playing

KNOWLEDGE-APL

KNOWLEDGE-APL is a board game for any number of players.
The Play consists of rolling dice, moving marker the
corresponding number of spaces as indicated on the dice,
and trying to acquire the most DEEDS to KNOWLEDGE about
APL. Each player competes for the most number of DEEDS
and the person with the most deeds when all deeds have
been distributed is declared the winner.

TO BEGIN:

Each player should pick a marker and place in on   ⛇ GO.
Roll the dice and the person with the highest roll plays
first.

TO PLAY:

The first player rolls the dice and mover his/her marker
the designated number of spaces. If the player lands on
an APL function (* ⌊ ⌈ | , ⍳ ⍴ ∈ ≤ = > -)
the player may challenge the computer (or other owner) for
this knowledge. The player must draw a card from the
ARGUMENT CARDS pile and compute (in less than one minute)
the result when arguments are supplied for the function on
which the player landed. Example: If player lands on ⌊
and draws the argument card:

$$K \leftarrow 4 \quad 6 \quad 8 \quad {}^{-}2$$
$$A \leftarrow 2 \quad 3 \quad 9 \quad 5$$

The player must report the result to $K \lfloor A$, Namely,

$$2 \quad 3 \quad 8 \quad {}^{-}2$$

If correct the deed to this function is given to the player.
If incorrect the deed remains with the computer.

If player lands on a branch command the player must draw a
card from the BRANCHING COMMANDS pile and do exactly what it
says. If the instruction sends player to another APL
function she may challenge computer (or other owner) for it.

If roll of dice lands you on ENDLESS LOOP you may stay
there as a visitor until next turn. Place marker in
JUST VISITING space.

If player lands on   → ENDLESS LOOP: Player must stay there
for next two turns unless she has a 'get out of endless
loop free' card.

If player lands on her own property she must respond
correctly or risk losing property to the computer.

PLAY PASSES TO THE LEFT:

The second player, and then each in turn, rolls the dice and
competes for ownership of APL knowledge. The play procedes
as above with the following addendum:

ADDENDUM:

The subsequent rolls of the dice may land a player on another
person's property (knowledge). The player may challenge the
owner for this knowledge (property). The player draws an
argument card and responds as she thinks the computer would
respond. (In less than one minute).

If response is CORRECT and the owner says it is CORRECT,
nothing happens...The player has a nice visit on the owner's
property. Play passes to next player.

If response is INCORRECT and the owner says it is CORRECT,
the deed must go back to the computer. Play passes to next
player.

If response is CORRECT and owner says it is INCORRECT, the
owner must give the deed to this knowledge to the player
challenging. Play passes to next player.

If response is INCORRECT and owner says it is INCORRECT,
player must give owner one of her previously acquired deeds
to some piece of APL knowledge.

TO FACILITATE MATTERS:

Someone should be available to type in arguments and
functions into the computer. This role may be shared by all
players.

The computer should serve as arbiter in all arguments
resulting from play.

REWARDS AND PENALTIES:

If a player rolls doubles she may take another turn. If
she rolls doubles again she must go to ENDLESS LOOP and
miss her next two turns, unless she has a 'get out of
endless loop free' card.

RULES FOR PLAYING

APL/500

DESCRIPTION OF GAME:

APL/500 is a game for 2 or more players. In playing this
game, the first player (first player is determined by the
highest roll on the dice) rolls the cubes out on the table.
The resulting symbols facing upward on the cubes are the
RESOURCES for that play of the game.

The player then draws a RESULTANT CARD and sets the one
minute timer. The player must try to form the longest
syntactically correct APL expression that will yield a
result within the range on the RESULTANT CARD.

SCORING:

If the player's expression is unchallenged——results within
the range on resultant card and within the time limit——
the player is awarded 5 points for every cube used to form
expression.

CHALLENGING:

To challenge an expression another player must shout ERROR
and explain what type of error report the computer will give
or the player must shout RESULTANT INCORRECT if player has
evaluated the expression incorrectly.

If the challenge is correct the player challenging gets 50
points for the correct challenge. If the challenge is
incorrect, the player subtracts 25 points from her score.

IMPOSSIBLE EXPRESSIONS:

If a player announces that no APL expression can be formed
to yield the resultant within the stated range, the time is
reset and all other players try to see if an expression can
be found. If no expression can be formed, the player first

announcing the impossibility gets 60 points for that play.
If an expression can be found, all players who do so will
get 60 points added to their score. Any challenges in this
play follow rules for challenging stated above.

WINNER:

The winner is determined by the person who gets 500 points
first.

ADDITIONAL POINTS ABOUT PLAYING APL/500:

Numbers may be formed by placing cubes side by side in the
following fashion:
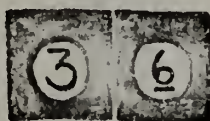
This is the number 36

Negative numbers may be formed by placing the minus sign
slightly above the other cubes in the following fashion:
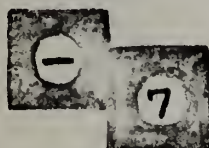
This is the number ¯7

A line has been drawn under 6, 9, ⌊ , and ⌈ to
distinguish these symbols.

INDEX

TERMS	PAGE

The following is a photo-reduced copy of the
actual manual used in this study.

APL
▼
FOR
[1]
TEACHER—
[2]
LEARNERS
[3]

[4]

[5]   ▼
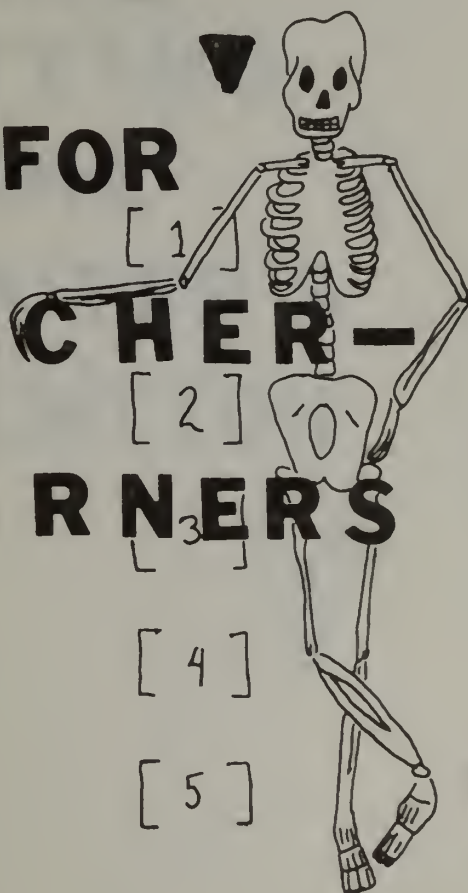
CHAPTER 1

GETTING
THE
COMPUTER'S
ATTENTION

COMMUNICATING
WITH A
COMPUTER

TELEPHONE

COUPLER

KEYBOARD
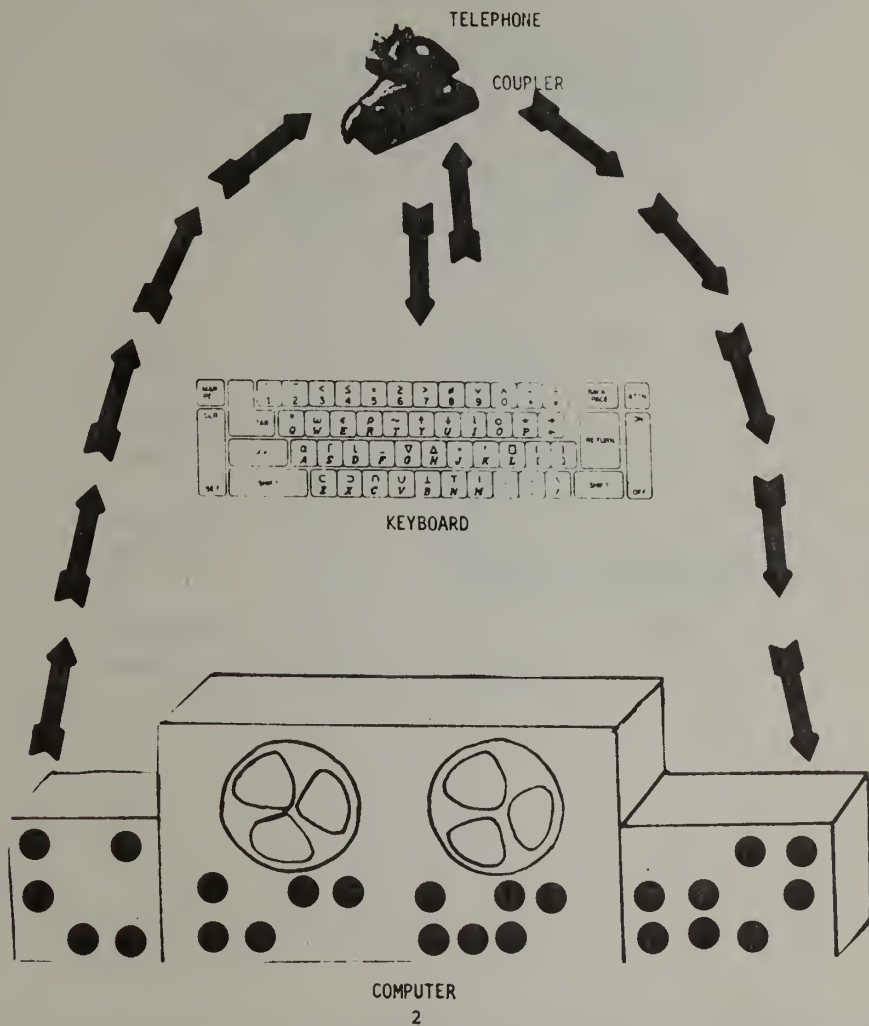
COMPUTER
2

SIGN—ON

(A ONE ACT PLAY)

CHARACTERS:

THE COMPUTER PROGRAMMER
THE COMPUTER TERMINAL (AN OBEDIENT SERVANT)

SETTING:

A ROOM IN THE SCHOOL OF EDUCATION

THE PROGRAMMER ENTERS THE ROOM WITH A DESIRE
TO BEGIN PROGRAMMING. A COMPUTER TERMINAL,
A TELEPHONE, AND A COUPLER ARE IN A CHEERFULLY
LIT ROOM EAGERLY WAITING FOR THE PROGRAMMER
TO PUT THEM TO USE.

PROGRAMMER: (TURNS ON THE SWITCH ON THE COUPLER
AND PICKS UP THE TELEPHONE)
(DIALS)   5 - 1 6 1 1
(WAITS TO HEAR HIGH PIERCING SOUND, THEN
PUTS TELEPHONE RECEIVER IN COUPLER)

(TYPES)   L

(PRESSES)   CARRIAGE RETURN

COMPUTER: (PRINTS)   USER NO.\

PROGRAMMER: (TYPES)   S2399

COMPUTER: (PRINTS)   C O D E
▮ ▮ ▮ ▮

PROGRAMMER: (TYPES)   EDOC

COMPUTER: (PRINTS)   TERMINAL 064   PORT 013
TIME(,DATE 9/8/73. OFF AT 1(3

PROGRAMMER: (TYPES)   APL

COMPUTER: (PRINTS)   CLEAR WS

3

SIGN–OFF

(A ONE ACT PLAY)

CHARACTERS:

    THE COMPUTER PROGRAMMER
    THE COMPUTER TERMINAL (AN OBEDIANT SERVANT)

SETTING:

    A ROOM IN THE SCHOOL OF EDUCATION
    THE PROGRAMMER HAS JUST COMPLETED HIS PROGRAMMING
    TASK.

PROGRAMMER: (TYPES)   )LEAVE

             (PRESSES)   CARRIAGE RETURN

COMPUTER: (PRINTS) TIME(SEC.

PROGRAMMER: (TYPES)   BYE

             (PRESSES) CARRIAGE RETURN

COMPUTER: (PRINTS) CPU USAGE
                       CONN HRS.

                    OFF AT 9/8/73

PROGRAMMER: (TURNS OFF THE SWITCH ON THE COUPLER AND
             HANGS UP THE TELEPHONE.)

             (TURNS OFF THE SWITCH ON THE KEYBOARD.)

4

CHAPTER 2

INTRODUCING

A

PROGRAMMING

LANGUAGE
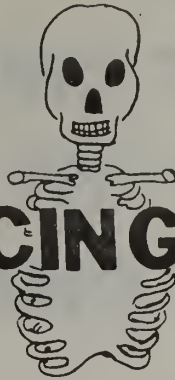
'HELLO' CARRIAGE RETURN

HELLO

'WELCOME TO'

WELCOME TO

'APL LAND'

NOTE: ● THIS SYMBOL ASKS
THE USER TO INDICATE COMPUTER'S RESPONSE.

6
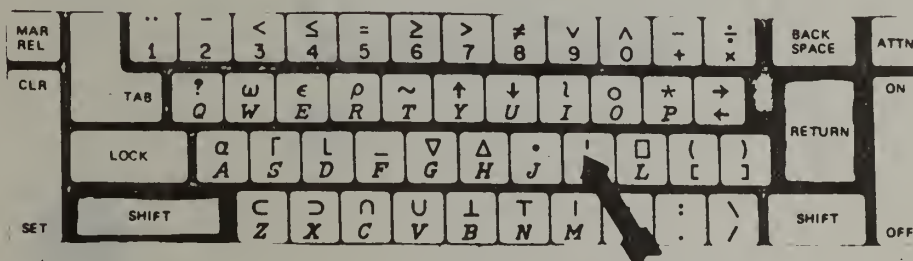
REMEMBER CARRIAGE RETURN

C ← 'TREE'

NOTE: THIS IS READ "STORE IN VARIABLE C, THE WORD TREE."

C

TREE

W ← 'PLUM'

W

PLUM

K ← 'DIME'

K

V ← '7-3'

V

G ←━ **7-3** REMEMBER <u>CARRIAGE</u> <u>RETURN</u>

G

4

J ←━ **19x2**

J

A ←━ **27÷9**

A

8
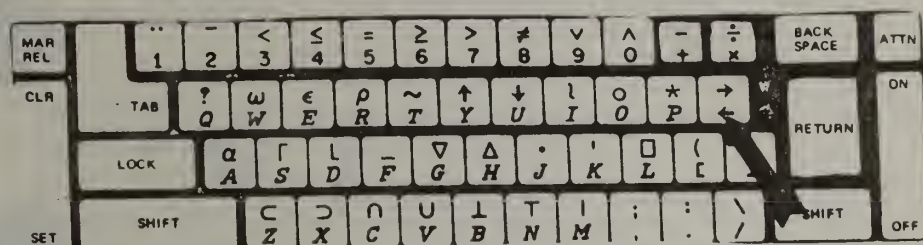
W
PLUM
C

W,C
PLUMTREE
C,W

K,C

G,J

9

V

V ← 1 2 3 4

V

Y ← 5 6 7 8 9

V,Y

Q

VALUE ERROR
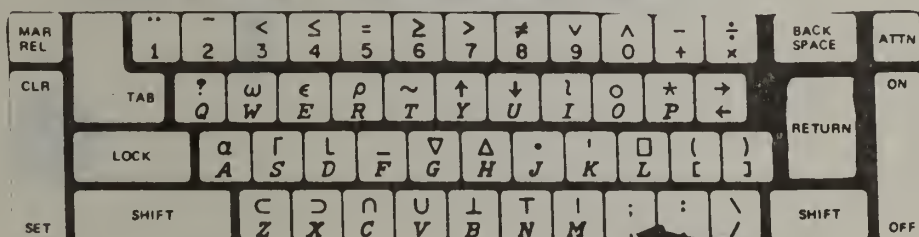
REVIEW

'FOOTBALL'

B ⟵ 'BACK'

H ⟵ 'HALF'

H, B

S

Z ⟵ 32 + 64

Z

11

CHAPTER 3

≥ < -=
★
PRIMITIVE +
L ÷ X
FUNCTIONS
I Γ

DYADIC

DYADIC -- dī•ad'ic, adj. HAVING TWO PARTS OR ELEMENTS.

LEARNING A NEW LANGUAGE REQUIRES NOT ONLY
MASTERY OF THE SYMBOLS, BUT ALSO THE MASTERY
OF THE SYNTAX OF THAT LANGUAGE.

THE
SYNTAX
OF
DYADIC FUNCTIONS

$$R \longleftarrow \blacktriangle D \blacksquare$$

D　IS ANY DYADIC FUNCTION

R　REPRESENTS THE RESULTANT

▲　IS THE LEFT ARGUMENT

■　IS THE RIGHT ARGUMENT

13

DYADIC
ARITHMETIC FUNCTIONS

$$9+5$$
$$14$$

$$3.82 + .1506$$

$$6-4$$

PRESS SHIFT KEY TO GET MINUS SIGN

$$3-8$$
$$^-5$$

14
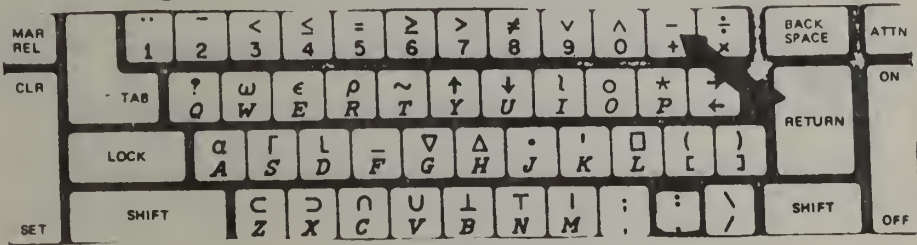
98.6 - 100

4 x 8.3

72 ÷ 8

1 ÷ 3

7 ÷ 0
DOMAIN ERROR
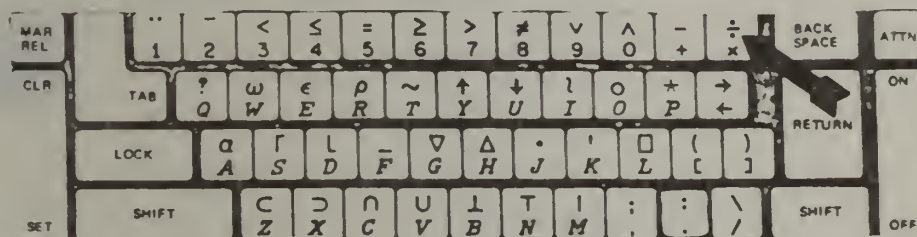
5+2 4 1 6 3

7 9 6 11 8

9-7 3 2 6

3x4 8

7 2 8 5-9 2 4 6

⁻2 0 4 ⁻1

5 3 x 7 21

16 9 25÷2 3 5

72 8 ÷ 9 1

5 7 - 3 1 2 8 4
LENGTH ERROR
9 4 7 3 + 5

2 9 x 12 8

7 6 4 81 ÷ 7 3

K ← 4

W ← 9

W

K + W

13

Z ← 2 3 4 5

K x Z

8 12 16 20

W - Z

18

DYADIC
RELATIONAL FUNCTIONS

3 < 7

1

5 < 4

0

8 < 25

9.2 < ⁻6

37 ≤ 22

0

5.4 ≤ 5.4

NOTE: THE RANGE OF THE RELA-
TIONAL FUNCTIONS WILL BE ZEROES
AND ONES.

19

0

$7 = 3$

$5 = 5$

1

$62 > 47$

$^{-}6 > 5$

$23 \geq {}^{-}23$

$0 \geq {}^{-}0.19$

$42 \neq 24$

1

20

6<4 5 6 7 8

0 0 0 1 1

6≥4 5 6 7 8

7 ⁻2 6=7 8 6

1 0 1

6 7 4 2≠6 3 8 1

9 8>1 8 2 9 7

LENGTH ERROR

21
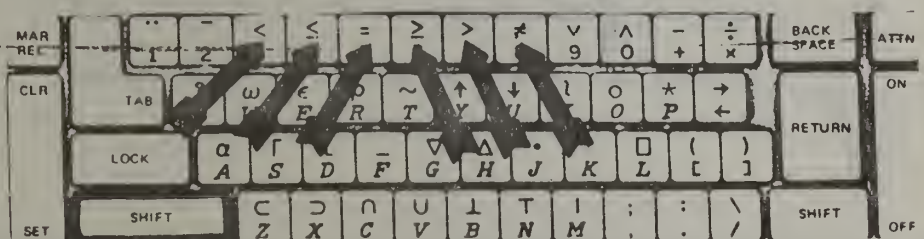
DYADIC
LOGICAL   FUNCTIONS

1 v 1

1

1 v 0

1

0 v 1

0 v 0

NOTE: THE DOMAIN AND RANGE
OF LOGICAL FUNCTIONS WILL BE
ZEROES AND ONES.
1 = TRUE
0 = FALSE

22

1∧1

1

1∧0
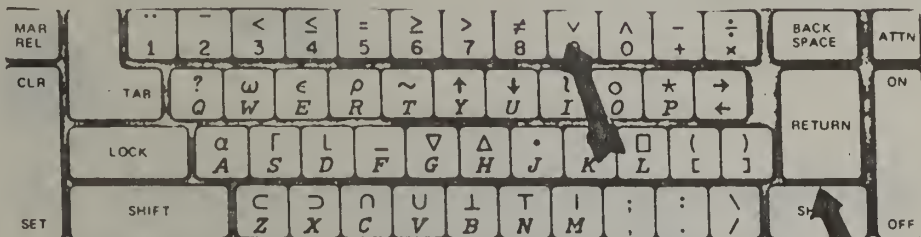
0

0∧1

0∧0

0 ~1
~0

NOTE: THIS IS A MONADIC
FUNCTION,(SEE CHAPTER 4)
BUT THE ~ NOT FUNCTION
IS USUALLY DISCUSSED WITH
THE ∧ "AND" AND THE ∨
"OR" LOGICAL FUNCTIONS.
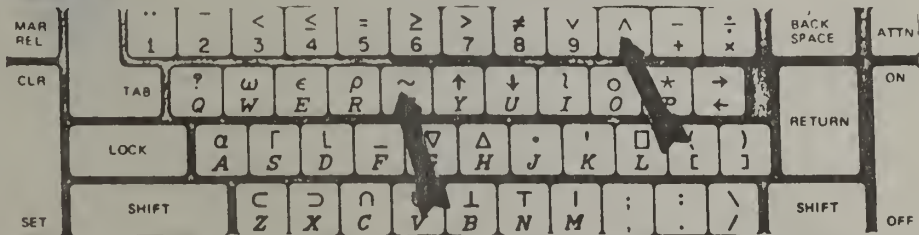
SYSTEMATIC
EXPERIMENTATION
OF   DYADIC FUNCTION

2★1

2

2★2

4

2★3

8

2★4

16

2★5

VARY RIGHT
ARGUMENT
SYSTEMATICALLY.

THEN...

24

3★2

9

4★2

16

VARY LEFT
ARGUMENT.

5★2

⬤

6★2

⬤

THEN...

VARY BOTH.

4★3

64

7★5

⬤

25

3★1 2 3 4   NOW ...

3 9 27 81   EXTEND OVER LISTS

5★1 2 3 4 5

2★¯2 ¯1 0 1 2

4★1 1.5 2 2.5

26
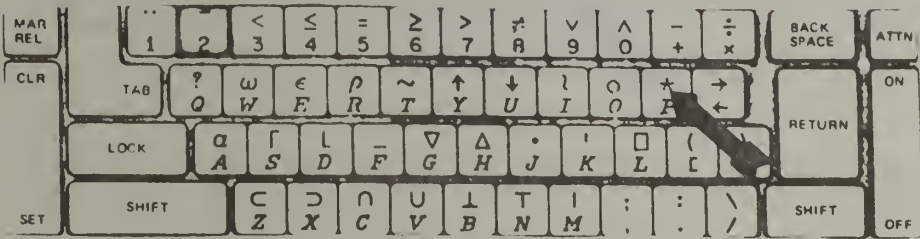
EXPERIMENT

DETERMINE BY SYSTEMATIC EXPERIMENTATION THE
MEANING OF THE FOLLOWING THREE FUNCTIONS (WHOSE
LOCATIONS ON THE KEYBOARD ARE IDENTIFIED BY
ARROWS):

27
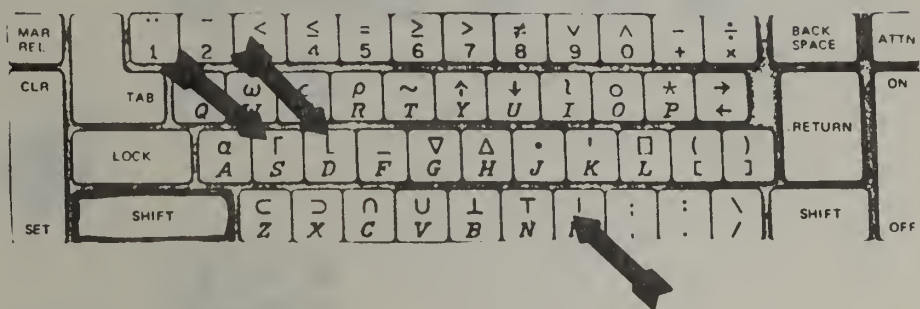
REVIEW

K

Z

K★Z

Q⟵5 3 1 8

KIQ

Q≧Z

KГQ

P⟵7 4 9 8 2

K+P

Q, P

ZLP

25 70 54÷8 7 6

29

CHAPTER 4

+ L !
÷ PRIMITIVE
x
? FUNCTIONS
Γ Γ I

MONADIC

MONADIC -- MŌ•NAD´IC, ADJ. HAVING ONE PART OR ELEMENT...

THE
SYNTAX
OF
MONADIC FUNCTIONS

R◄─◄ M●

M IS ANY MONADIC FUNCTION

R REPRESENTS THE RESULTANT

● IS THE RIGHT ARGUMENT

31

MONADIC FUNCTIONS

$$+8$$
$$8$$

$$+914$$
$$914$$

$$+67$$

$$+{}^-2$$

-4

⁻4

-76

⁻5.208

5.208

⁻.017

33
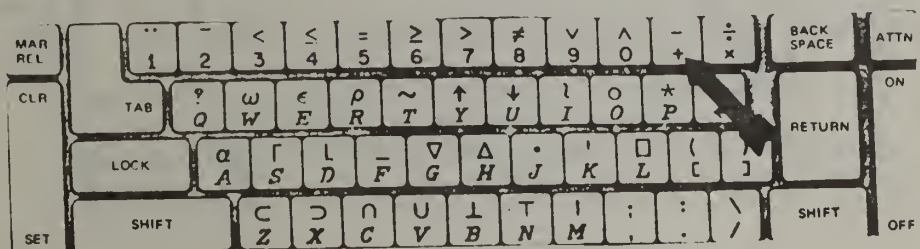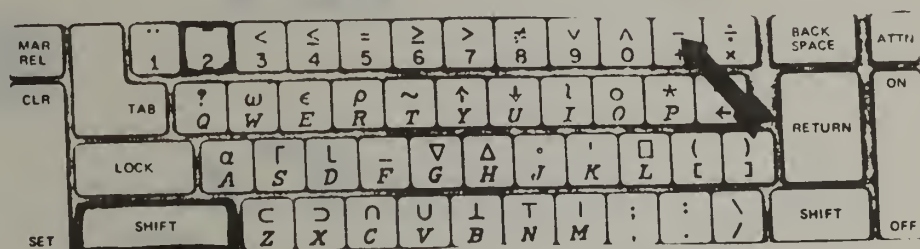
x6

1

x287

1

x4.593

1

x703

x.09

x⁻3

⁻1

34

x⁻7.55

x0

÷3

.3333

÷⁻2

⁻0.5

÷7

REMEMBER TO PRESS SHIFT KEY

NOTE: THE MAXIMUM NUMBER OF
DIGITS PRINTED TO THE RIGHT
OF THE DECIMAL POINT IS DETER-
MINED BY ENTERING A COMMAND OF
THE FOLLOWING FORM:

      ,DIGITS 4

OTHERWISE THE COMPUTER PRINTS
THE FIRST TEN SIGNIFICANT DIGITS.
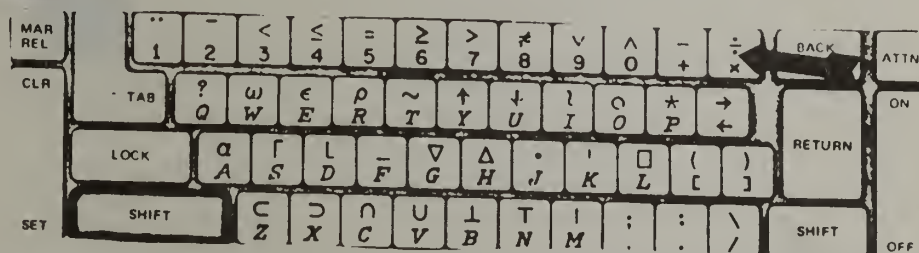
35

+29 ⁻4 5.6

29 ⁻4 5.6

+6.6 0 ⁻7

-9 ⁻5 8.26 0

x ⁻2 0.19 0 51

÷4 ⁻6 .11

36

SYSTEMATIC
EXPERIMENTATION
OF   MONADIC   FUNCTIONS

!1

1   USE POSITIVE RIGHT ARGUMENTS

    !2

2

    !3

6

    !4

24

    !5

37

SYSTEMATIC
EXPERIMENTATION
OF   MONADIC   FUNCTIONS

!1

1   USE POSITIVE RIGHT ARGUMENTS

    !2

2

    !3

6

    !4

24

    !5

37

!6

!8

!0 7 9

NOTE: ! IS FORMED BY TYPING A
QUOTATION MARK, BACKSPACE, THEN
THE PERIOD

38

POSITIVE INTEGERS

ι1

1

ι2

1 2

ι3

1 2 3

ι4

ι5
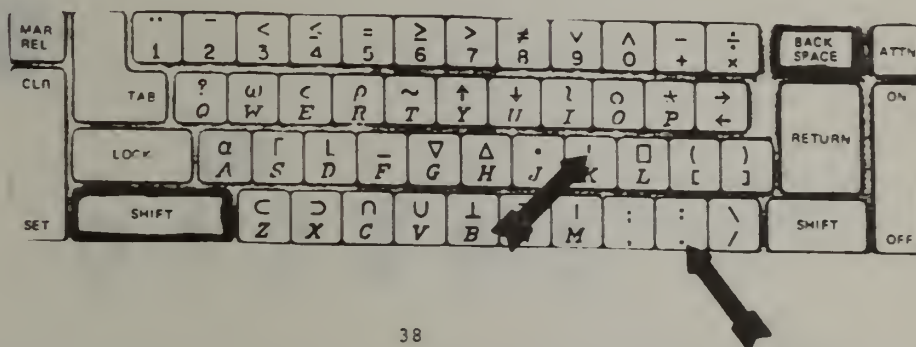
ι6

0 1 2 3 4 5

NOTE: THE ORIGIN IS MADE
ZERO BY THE FOLLOWING
SYSTEM'S COMMAND:

)ORIGIN 0

ι10

ιABABA'

ι⁻4

NOTE: THE DOMAIN OF THIS FUNCTION
IS THE SET OF POSITIVE INTEGERS.
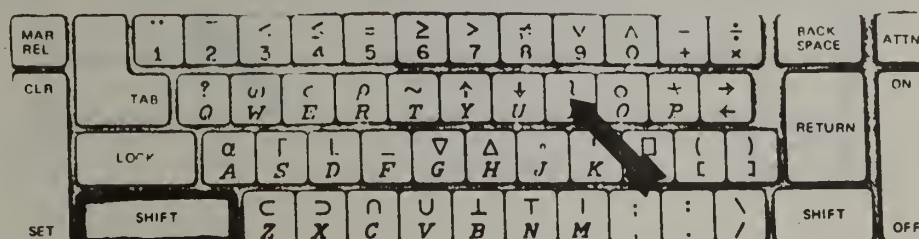
ι5 6

40

EXPERIMENT

DETERMINE BY SYSTEMATIC EXPERIMENTATION THE
MEANING OF THE FOLLOWING FUNCTION:

?.

NOTE: IN THIS EXPERIMENTATION
IT WILL BE NECESSARY TO USE ONLY
POSITIVE INTEGERS. YOU MAY USE
NEGATIVE AND FRACTIONAL RIGHT
ARGUMENTS TO SEE WHAT THE COMPUTER
WILL DO.

41
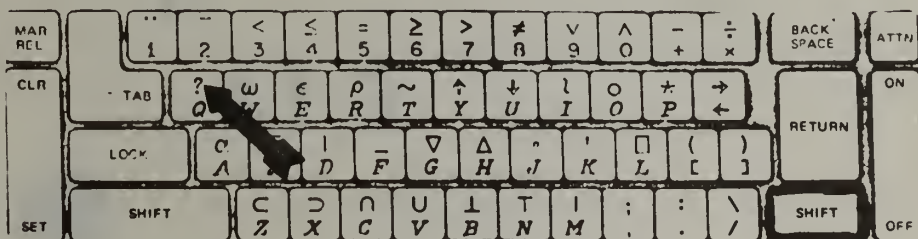
MORE
EXPERIMENTATION

L2.5

2

L2.2

2

L1.9

1

L1.6

1

L1.3

USE POSITIVE AND NEGATIVE,
FRACTIONAL AND INTEGRAL
RIGHT ARGUMENTS

L1.0

L0.9

L⁻0.9

⁻1

L⁻1.0

L⁻1.3

⁻2

L⁻2.3
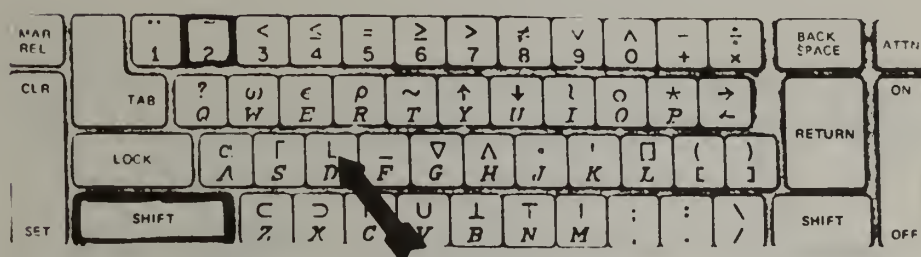
43

L8.0 8.6   9.2

L¯3.14 ¯7.8 ¯5

44

EXPERIMENT

DETERMINE BY SYSTEMATIC EXPERIMENTATION THE
MEANING OF THE FOLLOWING FUNCTIONS:

$$\lceil \bullet \qquad \lfloor \bullet$$

NOTE: IN THIS EXPERIMENTATION
IT WILL BE NECESSARY TO USE
POSITIVE AND NEGATIVE, AS WELL
AS, FRACTIONAL AND INTEGRAL
RIGHT ARGUMENTS.

45

REVIEW

H ←→ ⁻3.4 7 0.8

ΓH
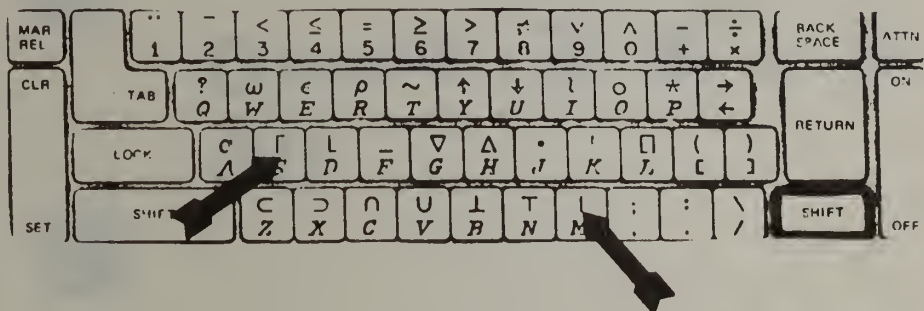
I ⁻91   2.351 0 ⁻5.1

xH

D ←← 4 2 8 1 5

ΡD

-H

÷D

!D

LH

+H

47

CHAPTER 5
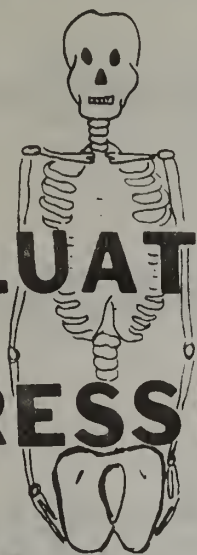
EVALUATING

EXPRESSIONS

RIGHT-TO-LEFT EXECUTION

R–L RULE

5+9−2x3

8

36÷3+8−2

4

(36÷(3+(8−2)))

4

20÷10−5+1

●

(20÷(10−(5+1)))

●

7−5x8+
SYNTAX ERROR

49

$$(2\star(3\times(7-6)))$$

●

$$56\div8L5+4$$

●

$$L10\div5I8$$

●

50
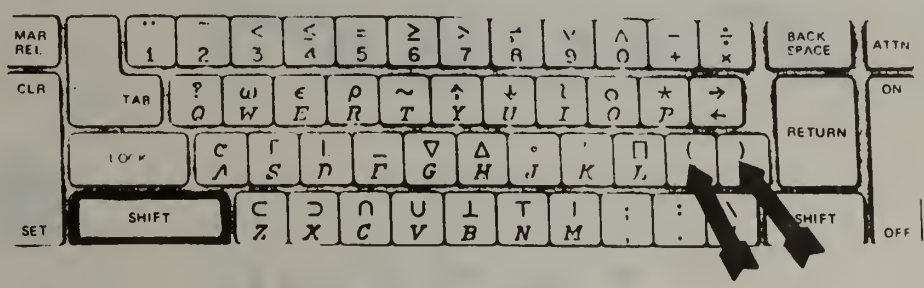
PARENTHESES PRIORITY

$$^-7^{(4+3)\times5-6}$$

NOTE: THE PARENTHESES
INDICATE DEPARTURES FROM
RIGHT TO LEFT EXECUTION

$$8\times(2\star0)+5$$

$$(^-9L8)\div3+1^-6$$

$$10\times(2.5+.7$$
SYNTAX ERROR

51

REVIEW

6 + 17−2l5

Γ20 ÷ ⁻6L⁻5

(l9★.5)+14−8

72÷8x6★0Γ

((3+7)x 6.39)÷9

CHAPTER 6

⌊/ +/   −/ φ

FUNCTIONS

÷/ OVER ⌈/

ρ LISTS ×/

VECTORS
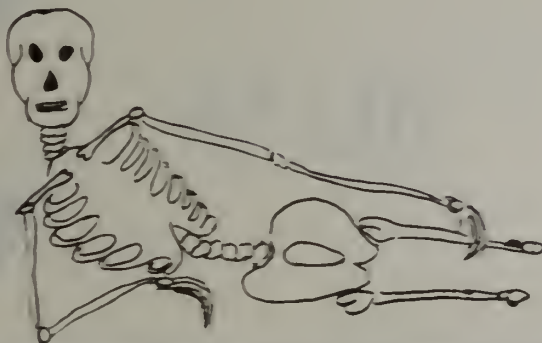
DYADIC FUNCTIONS
ON LISTS

+/9 4 7 5 4

29

+/⍳4

10

1+2+3+4

●

+/7 5 9.4

●

+/ ⍳10

●

K←━ 4 9 ‾5 8
K

●

+/K

●

−/K

●

4−9− ‾5− 8

●

(4−(9−(‾5−8)))

●

EXPERIMENT

EXPERIMENT WITH THE FOLLOWING FUNCTIONS OVER VARIOUS
LISTS:

$$\times/ \quad \lfloor/ \quad \lceil/ \quad \land/$$

$$\lor/ \quad </ \quad \div/ \quad |/$$

NOTE: TRY OTHER RELATIONAL FUNCTIONS TOO.

56
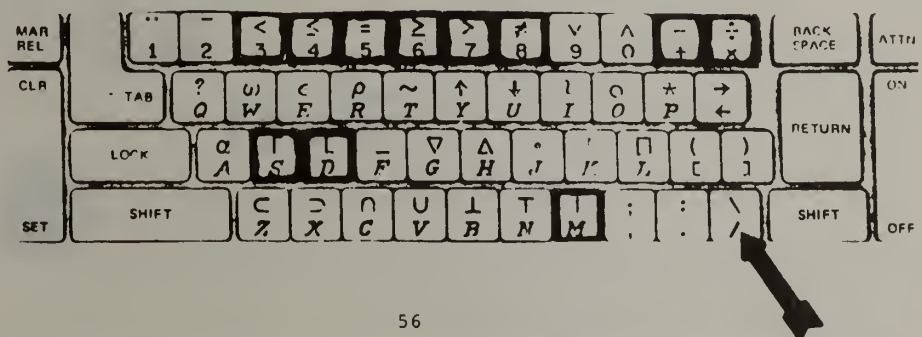
COMPRESSION

U←—1 0 1 1 0

F←—9⁻7 4 6.8 5

H←—0 1 0 0 0
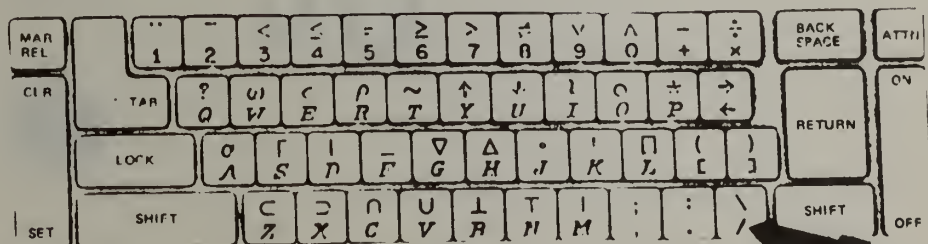
W←—'ABCDE'

U/F

9 4 6.8

0 1 0/7 5.6 9.47

5.6

H/F

●

57

U/W

ACD

H/W

H/F

0 0 0/9 2 4

1 0 0 0/7 4

58

EXPANSION

U ← 1 0 1 0 1

H ← 7 4 6

Z ← 6 8 4

S ← 1 0 0

W ← 'ABC'

U\H

7 0 4 0 6

U\W

A B C
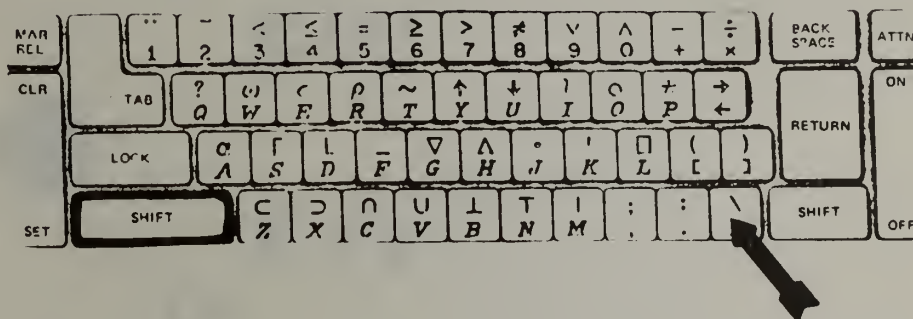
U\Z

●

F⟵ 'PIG'

A⟵ 7

U\F

S\A

MORE
DYADIC FUNCTION
ON LISTS (VECTOR RIGHT ARGUMENTS)

$$V \leftarrow \iota 5$$

$$V$$

●

$$1 \epsilon V$$

$$1$$

$$2 \epsilon V$$

$$1$$

$$5 \epsilon V$$

●

$$6 \epsilon V$$

$$0$$

61

8∈V

⁻7∈8 ⁻7 6 ⁻5 4
1
4 3 2∈8 4 2 9
1 0 1
⁻2 5∈5 6 7 8
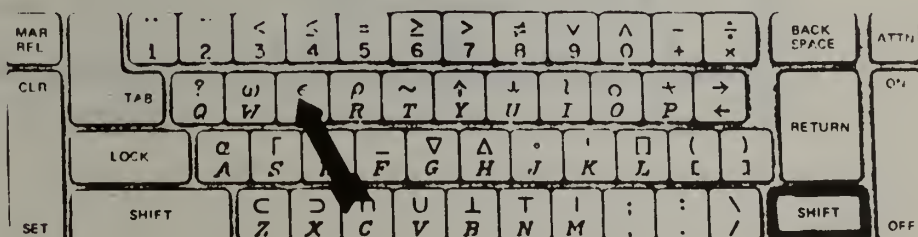
ՀՀ∈V

62
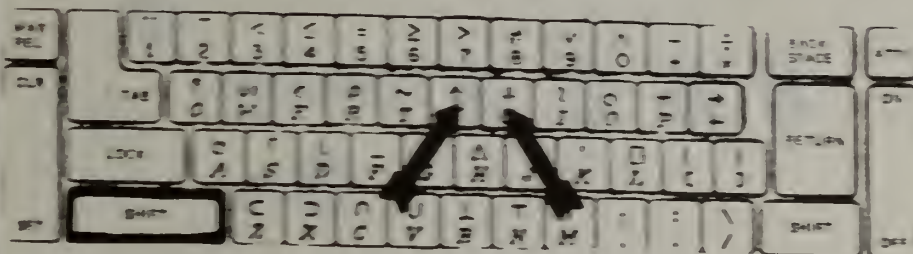
EXPERIMENT

MONADIC FUNCTIONS
ON LISTS

(VECTOR RIGHT ARGUMENTS)

V⟵ ⍳5

V

⍴V

5

Q⟵ ⍳6

⍴Q

6

B⟵ ¯5 3 6 8

⍴B

4

64

A ← 'COMPUTE'

ρA

7

G ← 4 ¯8 0 6 7.3

ρG

F ← 'APL IS FUN'

F

ρF

X ← 7

ρX

65

NOTE: THIS YIELDS A
BLANK

Y←'A BAT'
ρY

ρι12

Z←'C'
ρZ
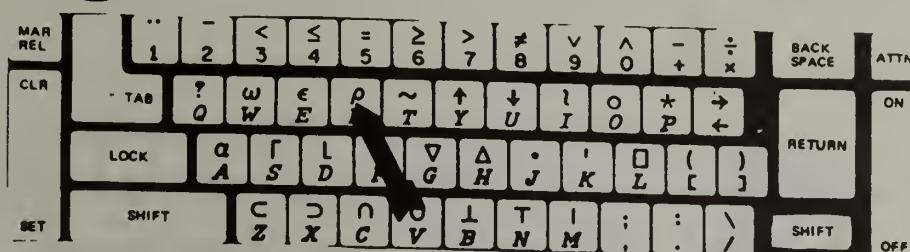
EXPERIMENT

DETERMINE BY SYSTEMATIC EXPERIMENTATION ON LISTS
THE MEANING OF THE FOLLOWING FUNCTION:

$$\Phi$$

NOTE: Φ IS FORMED BY TYPING O
BACKSPACE, THEN THE VERTICAL
LINE.
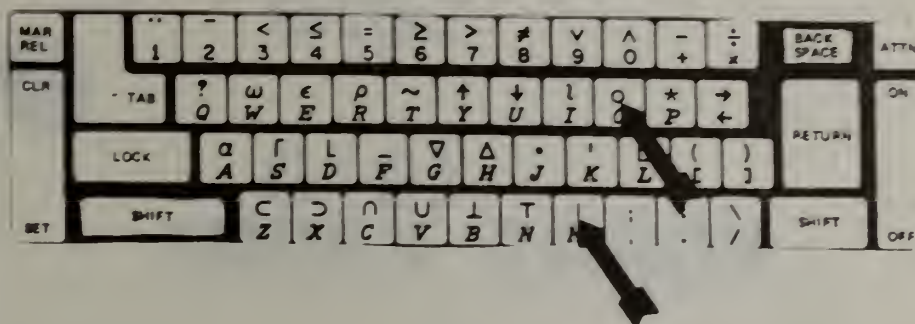
INDEXING
ON   LISTS
(VECTOR LEFT ARGUMENTS)

G

4 ⁻8 0 6 7.3

G[1]

4

G[2]

⁻8

G[3]

0

G[5]

G[6]
INDEX ERROR

68

G[⁻1]

●

G[2 5]
⁻8 7.3
G[ρ?5]

●

F[?6]
APL IS
F[2 1 5 10 8 9 3]

●

REVIEW

WHAT

HAVE

YOU

LEARNED?

MAKE uP YOUR OWN REVIEW. GIVE IT TO
A FRIEND TO TRY OUT.

70

CHAPTER 7
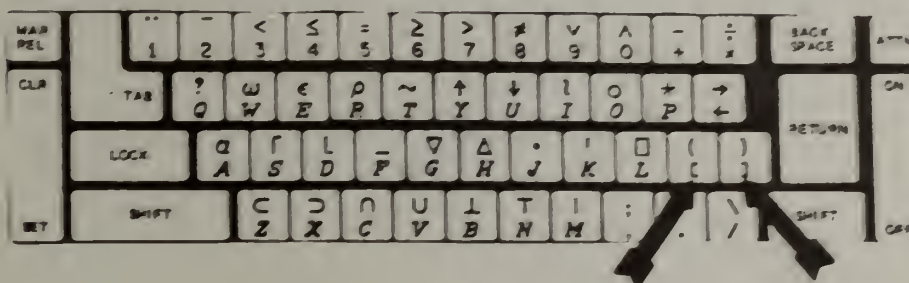
END ▼

DEFINED

FUNCTIONS

▼START
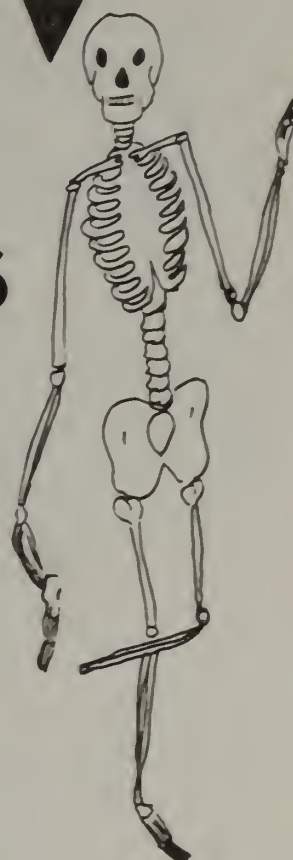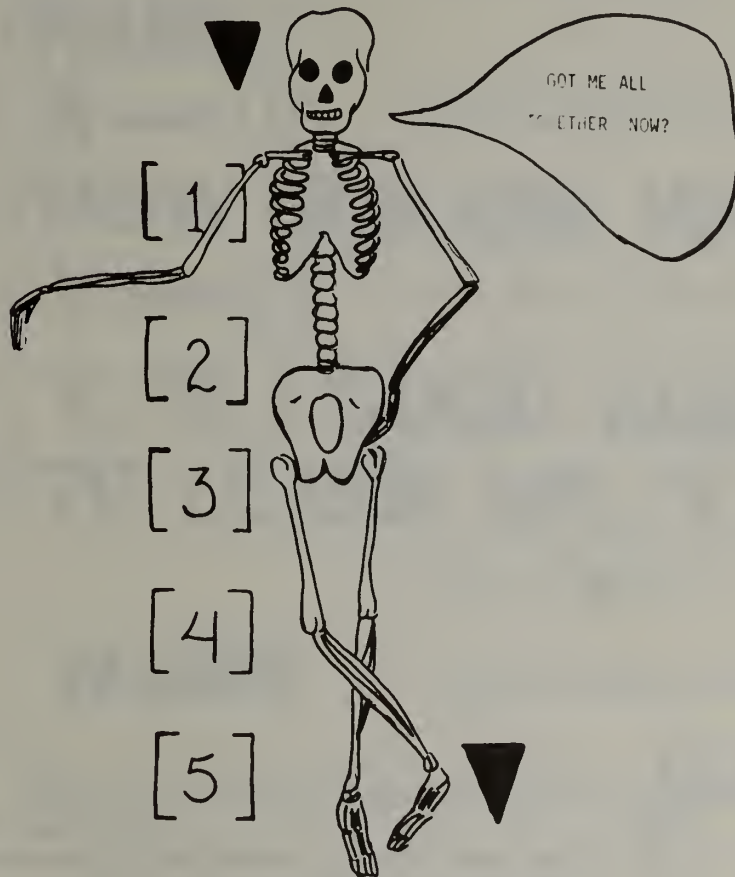
▽NAME

NOTE: ▽ DEL, STARTS AND ENDS ALL DEFINED FUNCTIONS.

[1] 'HELLO'

[2] 'WHAT IS YOUR'

[3] 'NAME?'

[4] X ⟵ ⎕

NOTE: ⎕ (TYPED ⎕, BACKSPACE, ') ACCEPTS LITERAL DATA

[5] 'HOW OLD ARE YOU?'

[6] Y ⟵ ⎕

NOTE: ⎕ ACCEPTS NUMERIC DATA

[7] Y; 'IS A GOOD AGE'

[8] 'TO LEARN APL;'X ▽

NOTE: ; AND , ARE USED TO SEPARATE LITERAL AND NUMERIC DATA

NAME

TYPE THE NAME OF THE FUNCTION, IN THIS CASE THE WORD NAME TO RUN THE PROGRAM

73
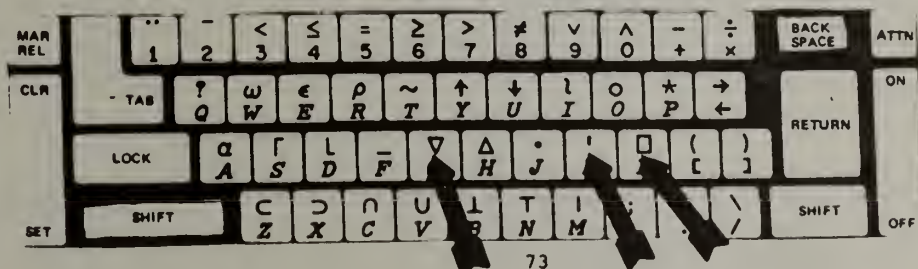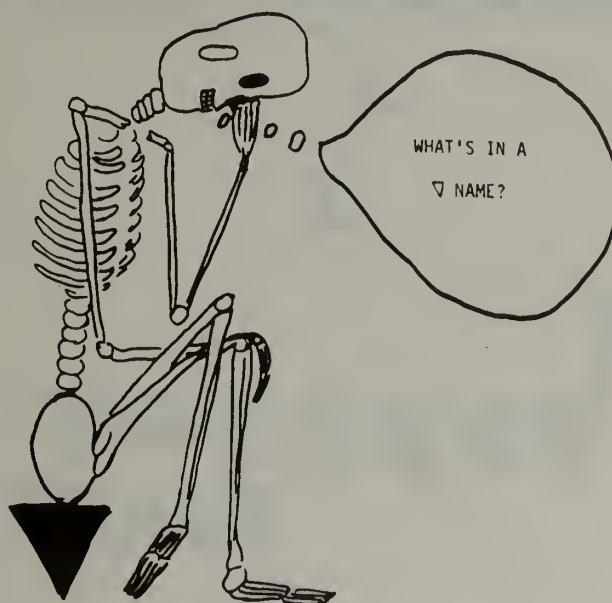
NILADIC

EXPLICIT RESULT

CUT ALONG THE DOTTED LINES

ANS ← AVERAGE

L

L

L ← 1 2 3 4 5

AVERAGE

3

L ← 2 4 6 8

AVERAGE

75

MONADIC
EXPLICIT RESULT

ANS←AVER L

L

L

▼

AVER 3 6 9 12

7.5

AVER 18 19 26

DYADIC
NO EXPLICIT RESULT

)ERASE AVE

L1 AVE L2

L1,L2

L1,L2

CUT ALONG THE DOTTED LINES

¯5 4 6 9 AVE 7 3

77
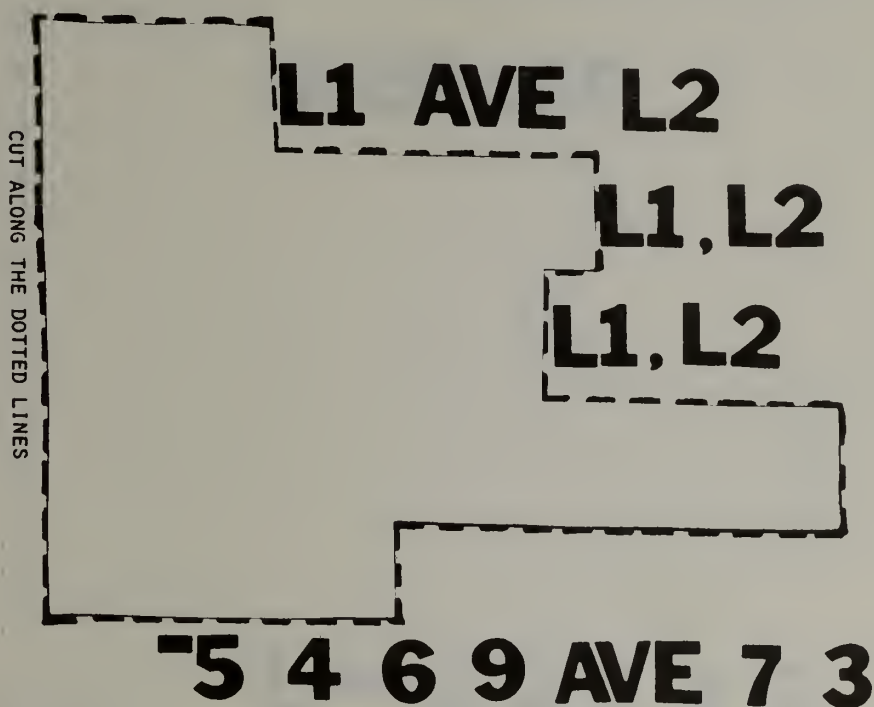
NILADIC
NO EXPLICIT RESULT

NOTE: WHEN EXPLICIT RESULTS ARE NOT
ASKED FOR IN THE HEADING YOU MUST ASK
FOR THE RESULTS TO BE PRINTED IN SOME
LINE OF YOUR PROGRAM

AVERAGE

L

L

CUT ALONG THE DOTTED LINES
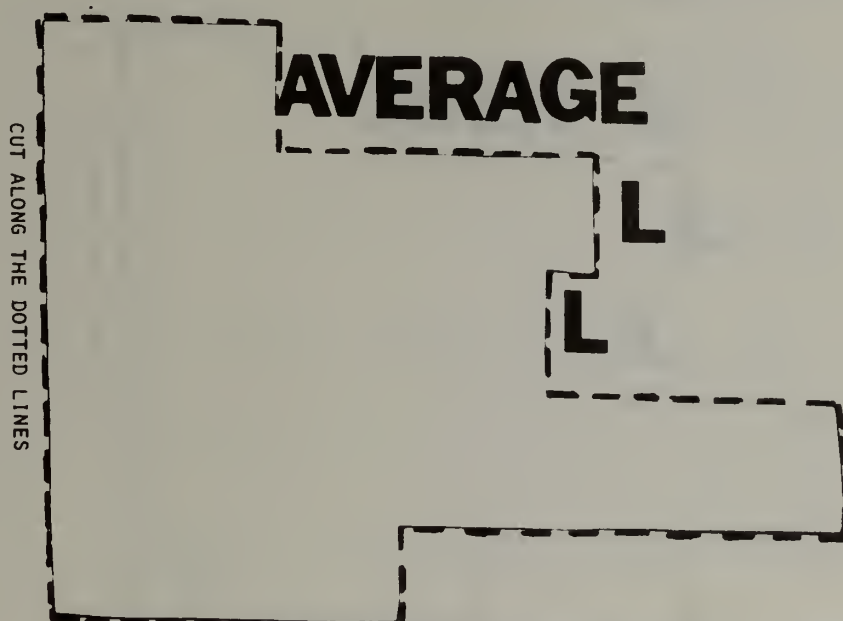
L←━1 2 3 4 5

AVERAGE

3

NOTE: IF YOU WISH TO CHANGE THE
PREVIOUS FUNCTION AVERAGE TYPE:

)ERASE AVERAGE

78

MONADIC
NO EXPLICIT RESULT

)ERASE AVER

AVER L

L

L

AVER 8.3 6 9.7

79

DYADIC
EXPLICIT RESULT

ANS←L1 AVE L2

L1, L2

L1, L2

▼
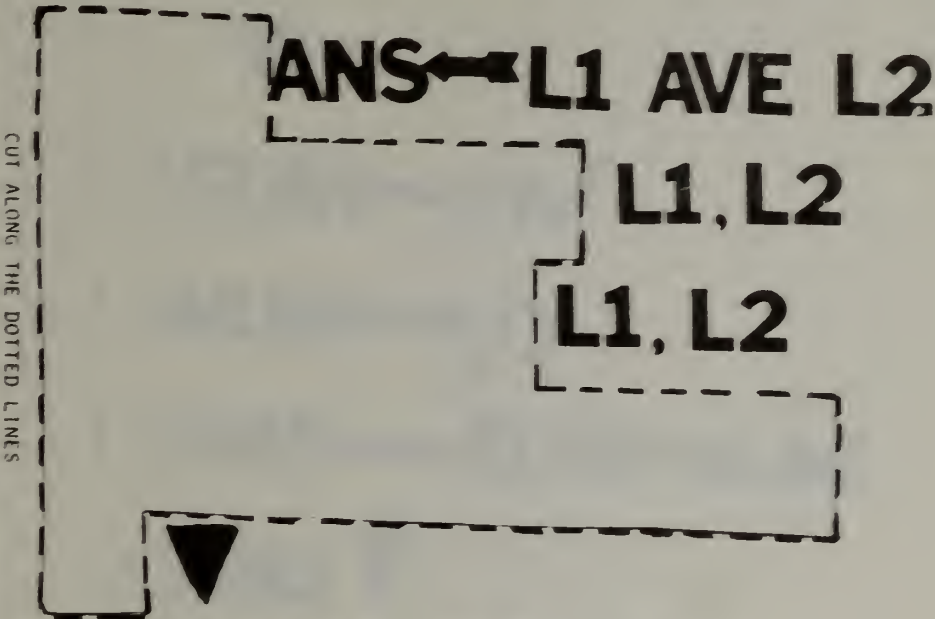
3 6 9 AVE 4 8

6

2 5 7 AVE 1 0 6

▼
[1]SUM←+/
[2]NUM←⍴
[3]ANS←SUM÷NUM
[4]ANS ▼

NO
BRANCHING
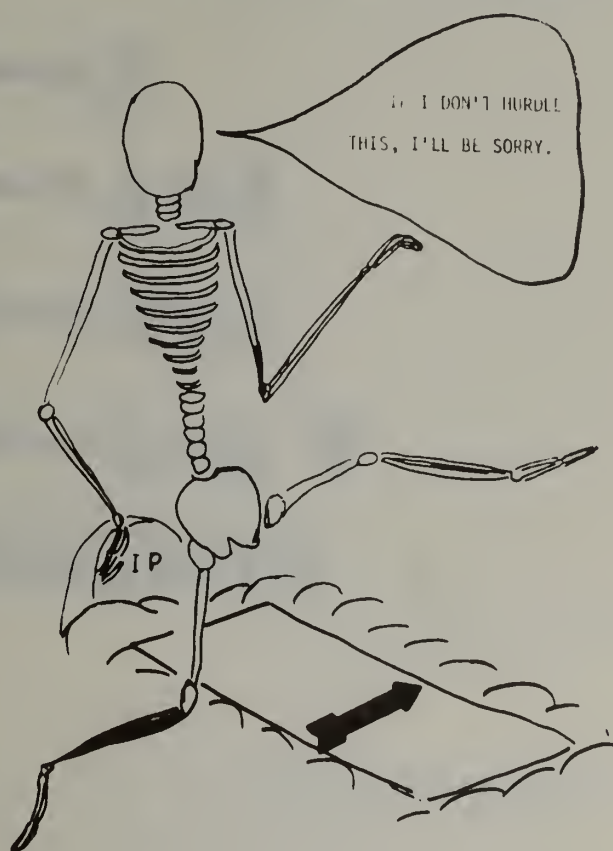
▼ Z ←— MULTIPLY

[1] Z ←— 3

[2] Z ←— 3+Z

[3] Z ←— 3+Z

[4] Z ←— 3+Z ▼

MULTIPLY

●

NOTE: THIS PROGRAM MULTIPLIES 3 x 4,
BUT IT DOES SO INEFFICIENTLY

UNCONDITIONAL BRANCHING

▼ Z◄━━MULTIES

[1] Z◄━━3

[2] Z◄━━3+Z

[3] ━━►2 ▼

MULTIES

NOTE: THIS PROGRAM IS IN AN ENDLESS
LOOP -- A PROGRAMMER'S NIGHTMARE! --
TO STOP THIS PROGRAM PRESS THE ATTN
BUTTON.

84
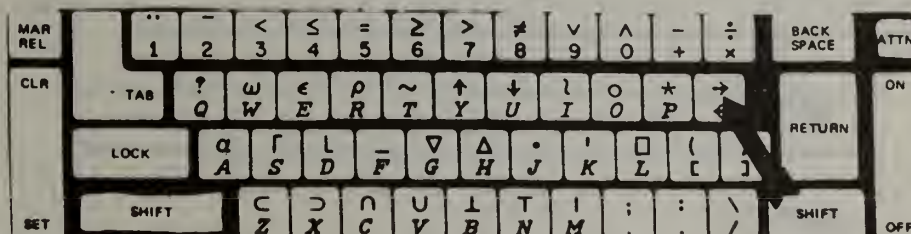
CONDITIONAL BRANCHING

▼ Z ← MULTI

[1] COUNT ← 1

[2] Z ← 3

[3] Z ← 3+Z

[4] COUNT ← COUNT+1

[5] → (COUNT<4)/3

[6] ▼

MULTI

CAN YOU GENERALIZE THIS
PROGRAM TO MULTIPLY ANY
NUMBER M BY ANY NUMBER N.

85

EDITING
DEFINED FUNCTIONS

DO I LOOK
ALL RIGHT?
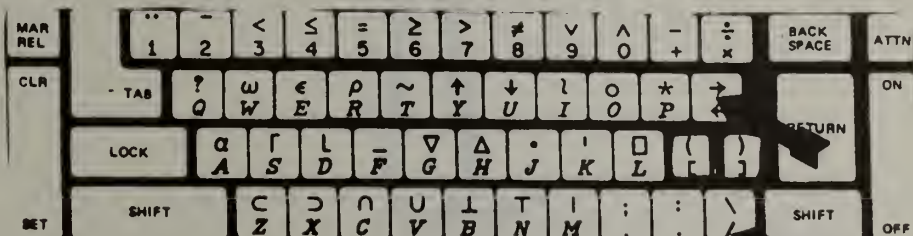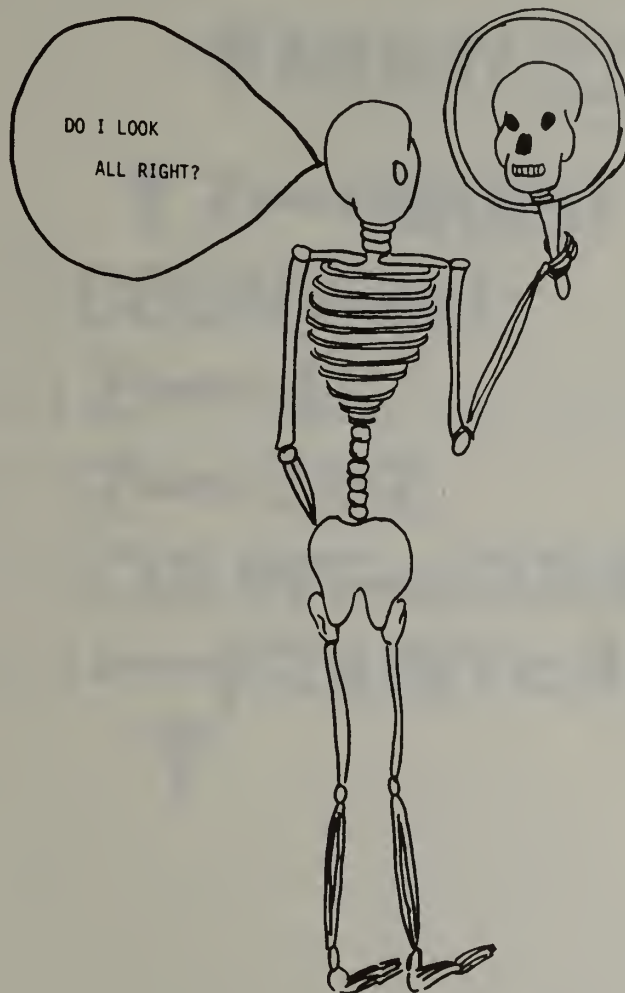
DISPLAY OF DEFINED FUNCTIONS

$$\nabla MULTI\ [\square]\nabla$$

$$\nabla\ Z \leftarrow MULTI$$
[1] COUNT ← 1
[2] Z ← 3
[3] Z ← 3+Z
[4] COUNT ← COUNT+1
[5] → (COUNT<4)/3
$$\nabla$$

CHANGE HEADER
OF
DEFINED FUNCTION

▼ MULTI [0□]

[0] Z⟵MULTI

[0] Z⟵MANYADS▼

NOTE: THE SECOND DEL MUST BE TYPED TO
LEAVE DEFINITION MODE.

88
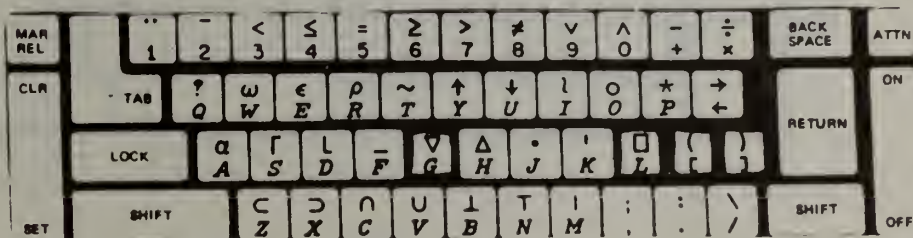
CHANGE LINE
OF
DEFINED FUNCTION

▼MANYADS[5□]

[5]➡(COUNT<4)/3

[5]➡3 ▼

89
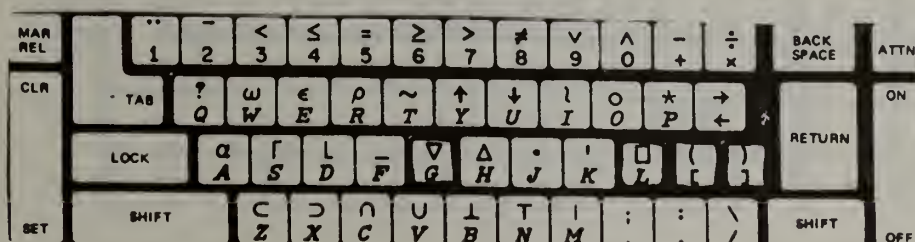
INSERT LINE

$$\triangledown MANYADS[3\square]$$

$[3]\ Z \longleftarrow 3+Z$

$[3]\ [2.5] \longrightarrow (COUNT=4)/0$

$[2.6]\ \blacktriangledown$
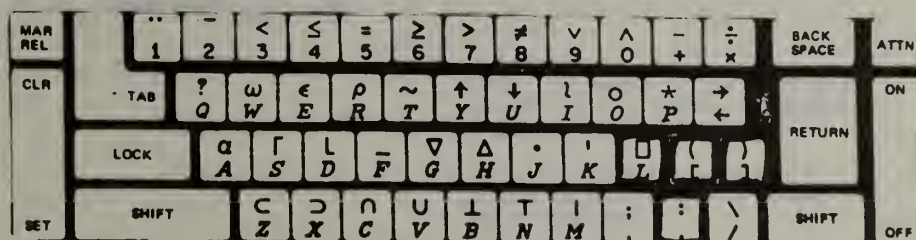
REVISED
PROGRAM

∇MANYADS[□]∇
∇Z←MANYADS
[1]COUNT←1
[2]Z←3
[3]→(COUNT=4)/0
[4]Z←3+Z
[5] COUNT←COUNT+1
[6]→3 ∇

NOTE: THE BRANCH TO ZERO INSTRUCTION
IN LINE [3] IS A COMMAND TO TERMINATE
THE PROGRAM.

NOTE: THE COMPUTER CONSECUTIVELY
RENUMBERS THE LINES OF THE PROGRAM
AFTER INSERTIONS.

EXPERIMENTATION

1. TRY TO REVISE THE MANYADS PROGRAM TO GENERALIZE THE MULTIPLICATION
   TO MxN. (YOU SHOULD CONSIDER THE SPECIAL CASES OF 0 x N, M x 0, AND
   0 x 0.) YOU MAY WISH TO CHANGE THE SYNTAX TO READ:

   $$Z \leftarrow M \quad MANYADDS \quad N$$

2. TRY TO WRITE A PROGRAM THAT DOES MANY MULTIPLICATIONS. WHAT
   PRIMITIVE FUNCTIONS DOES THIS NEWLY DEFINED FUNCTION REPLICATE?
   (REMEMBER SPECIAL CASES INCLUDING ZEROES)

3. TRY TO WRITE A PROGRAM THAT DOES MANY SUBTRACTIONS. WHAT
   PRIMITIVE FUNCTION DOES THIS NEWLY DEFINED FUNCTION REPLICATE?
   (REMEMBER SPECIAL CASES INCLUDING ZEROES)

92

REVIEW
THERE ARE SIX ERRORS IN THIS PROGRAM. CAN YOU FIND THEM?

▲P→B MANYM E

[1] COUNT→0

[2] P←1

[3] ←(COUNT=E)\0

[4] P←P x B

[5] COUNT←COUNT+1

[6] ←3▼

NOTE: COMPARE THIS PROGRAM WITH YOUR OWN MANY MULTIPLICATION PROGRAM.

93

CHAPTER 8

PROGRAM

PRODUCTION
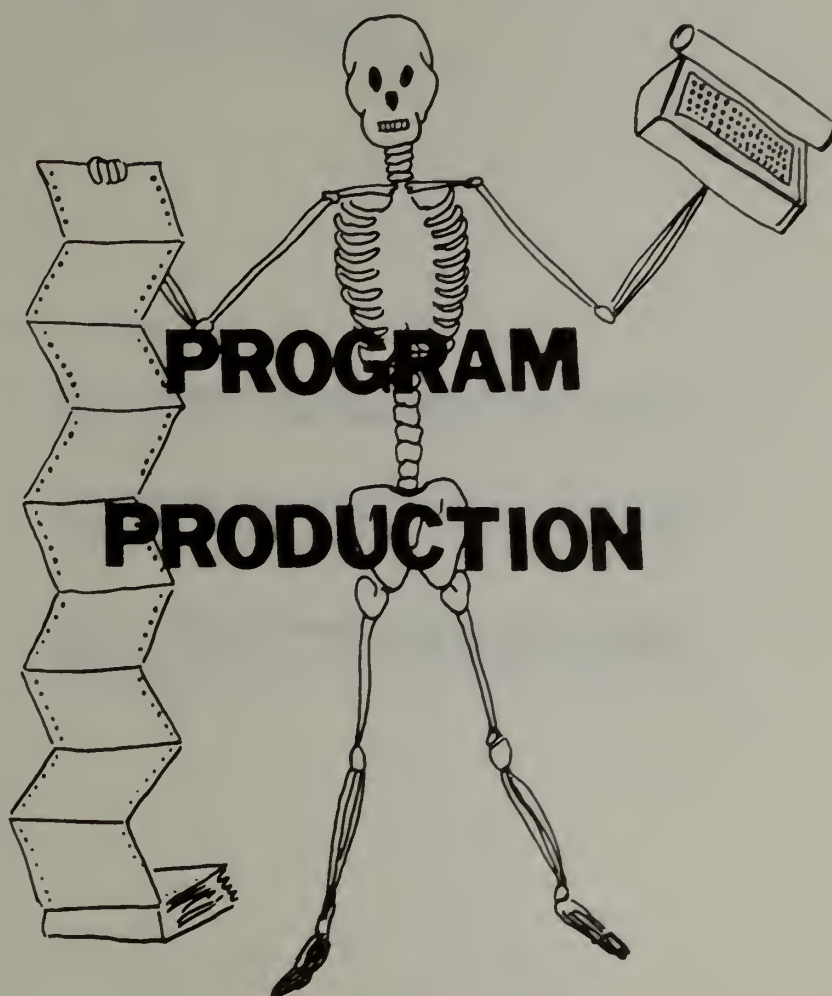
HERE IS A PROGRAM WHICH LISTS ALL THE FACTORS OF A NUMBER:

```
        ∇ DIVISORS←FACTOR NUMBER
[1]     ALL←ιNUMBER
[2]     REMAINDERS←ALL|NUMBER
[3]     EVENLY←0=REMAINDERS
[4]     DIVISORS←EVENLY/ALL
        ∇
```

USE THE FACTORS PROGRAM TO:

1. WRITE A PROGRAM TO DETERMINE IF A NUMBER IS PRIME. (A PRIME
   NUMBER IS A POSITIVE WHOLE NUMBER WHICH IS ONLY DIVISIBLE BY
   ITSELF AND ONE)

2. WRITE A PROGRAM TO PRODUCE PERFECT NUMBERS. (A PERFECT NUMBER
   IS ONE WHICH EQUALS THE SUM OF ALL ITS FACTORS EXCEPT ITSELF.
   6 IS THE FIRST PERFECT NUMBER:
                    $6 = 3 + 2 + 1$

3. WRITE A PROGRAM TO DETERMINE WHETHER TWO NUMBERS ARE RELATIVELY
   PRIME. (THIS MEANS THAT THE TWO NUMBERS HAVE NO COMMON FACTORS
   EXCEPT ONE.)

HERE IS A PROGRAM TO FIND THE GCD (GREATEST COMMON DIVISOR):

```
      ∇ GREATEST←N1 GCD N2
[1]     DIVISOR1←FACTOR N1
[2]     DIVISOR2←FACTOR N2
[3]     COMMON←(DIVISOR1∈DIVISOR2)/DIVISOR1
[4]     GREATEST←⌈/COMMON
      ∇
```

1. WRITE ANOTHER PROGRAM WHICH PERFORMS THE SAME TASK AS THE ABOVE
    PROGRAM.

2. WRITE A PROGRAM TO FIND THE LEAST COMMON MULTIPLE (LCM) OF TWO
    NUMBERS. (THE LEAST COMMON MULTIPLE IS THE SMALLEST NUMBER INTO
    WHICH BOTH OF THE GIVEN NUMBERS WILL DIVIDE EVENLY. THIS IS
    THE SAME AS FINDING THE LOWEST COMMON DENOMINATOR OF TWO FRACTIONS.)

3. WRITE PROGRAMS TO ADD, SUBTRACT, MULTIPLY, AND DIVIDE COMMON
    FRACTIONS. (FRACTIONS CANNOT BE EXPRESSED AS 3/4, BECAUSE THE
    SYMBOL "/" HAS ANOTHER MEANING. FRACTIONS CAN BE EXPRESSED AS
    TWO ELEMENT VECTORS: 3/4 IS 3 4). REDUCE ALL FRACTIONS TO
    LOWEST TERMS.

HERE IS A PROGRAM WHICH FINDS THE UNION OF TWO SETS:

```
        ∇ Z←A UNION B
[1]        Z←B,(~A∈B)/A
        ∇
```

STUDY THE ABOVE PROGRAM, THEN:

1.   WRITE A PROGRAM TO FIND THE INTERSECTION OF TWO SETS.

2.   WRITE A PROGRAM TO FIND THE COMPLEMENT OF A SET.

HERE IS A PROGRAM WHICH FINDS THE AREA OF A RIGHT ISOSCELES TRIANGLE:

```
        ∇ AREA←TRIANGLE S
[1]       AREA←S×S×1÷2
        ∇
```

USING THE ABOVE PROGRAM, TRY TO:

1. WRITE A PROGRAM TO FIND THE AREA OF A SQUARE USING THE TRIANGLE
    PROGRAM.

2. WRITE A PROGRAM TO FIND THE AREA OF A PARALLELOGRAM

3. WRITE A PROGRAM TO FIND THE AREA OF THE FOLLOWING HEXAGON:

HERE IS A PROGRAM WHICH SIMULATES THE "AND" FUNCTION IN
PORPOSITIONAL CALCULUS USING "ORS" AND "NOTS:"

```
        ∇ Z←A AND B
[1]       Z←~((~A)∨(~B))
        ∇
```

```
        1  ∧  1
1
        1  ∧  0
0
        0  ∧  1
0
        0  ∧  0
0
```

HERE IS THE TRUTH TABLE
FOR "AND":

WRITE A PROGRAM WHICH SIMULATES THE "IMPLICATION" AND THE
"IFF" FUNCTIONS IN PROPOSITIONAL CALCULUS.

HERE IS A PROGRAM WHICH ADDS NUMBERS (WRITTEN AS VECTORS) IN BASE TEN:

```
      ∇ SUM←N1 ADDITION N2
[1]      SUM←BASE|N1+N2
[2]      CARRY←⌊(N1+N2)÷BASE
[3]      →(∧/0=CARRY)/0
[4]      SUM←(0,SUM)ADDITION CARRY,0
      ∇
```

STUDY THE ABOVE PROGRAM, THEN:

1. WRITE A PROGRAM WHICH SUBTRACTS NUMBERS IN BASE TEN, THEN ANY BASE.

2. HERE IS AN ALGORITHM THOUGHT UP BY A SECOND GRADER WHICH SUBTRACTS
    NUMBERS IN BASE TEN. STUDY THE ALGORITHM, THEN TRY TO TRANSLATE
    IT INTO APL COMMANDS.

WRITE A PROGRAM TO CONVERT A NUMBER IN ANY BASE TO BASE TEN.

WRITE A PROGRAM TO CONVERT A NUMBER IN BASE TEN TO ANY OTHER BASE.

101

WRITE A PROGRAM TO CONVERT CENTIGRADE TO FAHRENHEIT DEGREES.

WRITE A PROGRAM TO CONVERT FAHRENHEIT TO CENTIGRADE DEGREES.

102

WRITE A PROGRAM WHICH WILL SHOW THE APPROXIMATE VALUE OF PI.
(PI EQUALS THE RATIO OF THE CIRCUMFERENCE OF A CIRCLE TO ITS
DIAMETER.)

$$\pi = c \div D$$

FIND MANY CIRCULAR OBJECTS, MEASURE THEM, AND ENTER THESE FIGURES IN
YOUR PROGRAM.

103

WRITE PROGRAMS TO REPLICATE THE FOLLOWING PRIMITIVE FUNCTIONS:

ABSOLUTE VALUE   (|)

FACTORIAL   (!)

EXPONENTIATION   (•)

MAXIMUM   (⌈)

GREATER THAN   (>)

PROBLEM

HOW MANY PAIRS OF RABBITS WILL BE PRODUCED IN A YEAR,
BEGINNING WITH A SINGLE PAIR, IF IN EVERY MONTH, EACH
PAIR BEARS A NEW PAIR WHICH BECOMES PRODUCTIVE FROM
THE SECOND MONTH ON?

(THIS CELEBRATED PROBLEM GIVES RISE TO THE "FIBONACCI
SEQUENCE" 0 1 1 2 3 5 8 13 21 ...)

WRITE A PROGRAM WHICH WILL SOLVE THE PROBLEM.

WRITE A PROGRAM WHICH WILL PRINT OUT THE FIRST TEN NUMBERS IN THE
SEQUENCE.

INDEX

CHAPTER V

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Summary

In Chapter I, the problem area——"disadvantaged" elementary school mathematics teachers———was defined and suggestions for revitalizing existing mathematics teacher training programs were given. These suggestions included:

1. strengthening link between methods and content components in teacher training programs;

2. developing pedagogical strategies to enrich curricula in the affective domain;

3. and, encompassing computing technology to improve understanding, pedagogical strategies and self-concepts.

The central theses underlying these revitalization suggestions were:

1. A programming language such as APL is well-suited as a conceptual framework for teaching mathematics concepts to elementary school teachers.

2. Programming heuristics are well-suited as "epistemological primitive" for mathematics teacher training courses.

3. A computer, in its role as "obedient servant" is the ideal pupil to help teachers clarify and organize their own thoughts and build self-confidences in mathematics and programming.

The philosophical underpinnings of these central theses were: (1) "A good way to learn something is to teach it" and (2) "Learning begets learning."

In Chapter II, a review of previous attempts at utilizing computing technology to teach mathematics, students, and teachers was discussed. A distinction was made between those CAI--computer-assisted-instruction projects which seek to control students and the CAI_1--computer-augmented-inquiry projects which seek to give control of technology to the user.

The next chapter was devoted to a description and an evaluation of the "Mathematics and Programming" (MAP) course taught by the author to prospective elementary school majors at the University of Massachusetts. The specific objective of the course were:

1. to provide prospective elementary school teachers with sufficient programming skills to permit them to articulate their thoughts about selected topics in mathematics to a computer via A Programming Language;

2. to suggest pedagogical strategies classroom teachers could use when introducing computer programming to elementary school students;

3. to explicate cognitive and affective behavioral objectives teachers should stress when programming techniques are employed in elementary schools.

The evaluative measure, though administered to too small a sample, yielded positive results worthy of further investigation.

Finally, in Chapter IV, the actual materials developed by the author and used in the MAP course were described. These materials included a manual titled —— "APL for Teacher-Learners" and a facilitators guide titled —— "Supplemental Guide to APL for Teacher-Learners." Taken in conjunction, these materials provided prospective teachers with APL programming concepts, topics in elementary school mathematics to program, and guideposts in the cognitive and affective domain to facilitate their teaching of topics in mathematics and programming to elementary school children.

The central theme permeating the preceding four chapters was the notion that learning A Programming Language may help in the development of heuristic reasoning and in the clarification of thought processes. With this theme in mind, we can use some programming terminology —— namely, systems commands in APL —— to facilitate our thinking about the education process in general.

> NOTE: An important feature of computer software is its systems commands. These commands are dictates to the system which allows users to manipulate data within given workspaces (See Chapter I of "Supplemental Guide to APL for Teacher-Learners" for explanation of workspace). (For instance, a command to save work being done at the terminal could be written)SAVE MYWORK, where MYWORK is the name of a workspace.)

The conclusion this document offers will be discussed in terms of systems commands:)ACCESS,)CONTINUE, and)SAVE, and the workspace names: TECHNOLOGY, COMPUTERS, and TEACHER/LEARNERS. These conclusions, based on evaluative responses of MAP participants, will outline ways in which school <u>systems</u> should be responding to the <u>commands</u> of individuals clamouring for self-identity, self-respect, and self-control.

Also, included in this chapter will be recommendations on ways of extending our knowledge about the effectiveness of computer technology in education.

<div align="center">

<u>Conclusion</u>

</div>

)ACCESS TECHNOLOGY

One "command" school systems must begin to respond to is the command by individuals to be given <u>access</u> to computer technology. Heretofore, computer technology has had access to students and teachers but the converse has not been true. With an increased amount of regularity, teachers and students in urban settings, particularly, are becoming aware of the computer's access to them. Enrollment, class assignments, report card distribution, and, yes, even some classroom instruction is being handled by the computer. But, teachers and students are merely the end-users of this technology. Arthur Luehrmann of Dartmouth posed the poignant question: "If the computer is so powerful a

resource that it can be programmed to simulate the instructional process, shouldn't we be teaching our students mastery of this powerful tool?"[1]

Many school boards and policy-makers have objected to this student-teacher accessibility to computers on the grounds of questionable cost-effective. Some stress the cost factor when giving arguments against computing technology in schools, while others emphasize the effectiveness question.

Looking at the effectiveness question for a moment, we find dissenters saying, "What is all this business about giving children computing power——many can't even add yet. . . . We can't support computer enrichment programs when compensatory (remedial) programs are what's needed." Or many will say, "If you can show me where this material is better than the old methods we learned by, then I might reconsider it."

Individuals attempting to answer these concerns can look to Alexander Gerschenkron's book titled Economic Backwardness in Historical Perspective to find an intriguing analogy to rebut arguments against uses of computers on the bases of effectiveness. Gerschenkron

[1]Luehrmann,"Should the Computer Teach the Student or Vice Versa?", p. 410.

noted that less developed nations aspiring for economic
self-control, insisting upon most modern technology
and refusing to go through all the "pre-requisites" to
industrial development, suffer initial phases of apparent
inefficiency but then spurt forward at a rate rapid
enough to gain significantly on more industrialized
nations.[2]

Similarly, it would seem that "disadvantaged"
teachers and students aspiring for self-respect and
self-control could profit from the inclusion of the most
modern computing technology in their schools. If CAI_1
programs which give power to users and remove stigmas of
remediation from materials, are implemented in urban and
rural schools where achievement scores and self-image
indices are reportedly low, major spurts in learning
competencies and self-concepts may be possible. Granted,
there may be initial misuses and minimal short-term
impacts, but in the long-run, learning competencies due
to technological accessibility, will certainly outweigh
initial ineffectivenesses.

)CONTINUE COMPUTERS

Looking at the cost side of the cost-effective coin
we come to another "command" to which individuals must

---

[2]Alexander Gerschenkron, Economic Backwardness in
Historical Perspective (New York: Frederick A. Praeger,
Inc., 1962), p. 40.

ask school systems to respond. Those opponents to computer
technology in school who feel that costs are prohibitively
high will need to be convinced of the necessity and the
feasibility of utilizing this technology.

A case was made by Felix Kopstein and Robert Seidel
for the feasibility of affording computers in schools by
showing a comparative cost analysis of computer costs and
per pupil expenditures.[3] But a case must also be made
for why we cannot afford not to have computers in all
schools. The necessities are these: In an already
dichotomized work-oriented society of "haves" and
"have-nots," the introduction of computers into some
elitist schools (as is being done) and not in other
schools will only serve to widen the schism. As
questioning individuals we must ask policy-makers "How
long will our children be employable and for what jobs
if elitist schools are turning out computer users by
the thousands?" and "How much longer will a computer
illiterate be considered educated?"[4]

---

[3]Felix F. Kopstein and Robert J. Seidel, "Computer-
Assisted Instruction," Computer-Assisted Instruction: A
Book of Readings (New York: Academic Press, Inc., 1969),
pp. 327-362.

[4]Luehrmann, "Should the Computer Teach the Student
or Vice Versa?", p. 410.

If these questions are answered with honesty and concern for succeeding generations, we must command continued and extended use of computers in all schools.

)SAVE TEACHER/LEARNERS

A final "command" individuals must insist school systems address themselves to is the command to save teacher/learners. Giving computing power to teacher/learners may be a necessary, if not sufficient, step to save teachers and learners from the school structures which inhibit the freedom to pursue learning in equal and creative partnerships. Computing power may also save teachers and learners from systems which manipulate them. And, finally, computing power may save teachers and learners from systems which repress and surpress intellectual curiosities.

School systems must respond to these "commands" by individuals for access to technology, for continued use of computer, and to save teacher/learners. In so doing, systems will then, and only then, begin to produce human beings who are self-starting, self-regulating, and self-respecting with capabilities of self-controlling and self-defining.

Recommendations

As the Survey of Literature (See Chapter II) indicated, very few studies have been done to investigate the possibilities of computer use in elementary mathematics teacher training programs. The need for additional work in this area is great. From experience and insights gained developing materials, teaching and evaluating the pilot and MAP course, this author is compelled to make the following recommendations.

1. The MAP course should be taught again with a larger sample of pre-service teachers.
2. The programming materials should be tried out on children ages 10-12 and on in-service teachers.
3. The content materials in the "APL for Teacher/Learners" should be extended to include more work in arrays and more topics in mathematics.
4. The content materials developed using APL should be written using other programming languages (i.e. LOGO, BASIC, etc.)

Observations made by this author during the MAP and pilot course have suggested designs for formal research which may get at the question of effectiveness of computer programming in teacher education. The following are a few specific recommendations:

1. A study should be conducted comparing the mathematics understandings of prospective elementary school teachers using programming with the mathematics understanding of teachers who have not used programming.

(This study could also be done using in-service teachers or students in its sample population.)

2. A study should be designed that would test the hypothesis which asserts a positive correlation between "mathematics understanding" and "programming articulation." (This study should be done on pre-service and in-service teachers and students.

3. A study should be done to determine the effectiveness of creative teaching techniques and programming on self-concept development and positive attitude development of prospective elementary school teachers (or in-service teachers, or students.)

4. A longitudinal study could be conducted to determine if students of teachers taught using the programming techniques to clarify concepts in mathematics understand these concepts better than students of teachers who have not been exposed to programming.

A Final Word

To deny others (students and teachers, alike) exposure to methods and content in computing technology or to continue to use traditional methods and out-moded materials to the exclusion of computing technology because you are more comfortable without its inclusion is tantamount to lobotimizing patients for comfort's sake, even though they are impaired for life.[5]

_____

[5] Thomas F. Pettigrew, "Racially Separate or Together?" Journal of Social Issues, XXV (November, 1969), 51, offers this comparison in terms of postponing the integration of public schools.

The task of preparing teachers to arrange environments in which children of tomorrow can become competent learners will first require a rejection of exclusion practices for comfort's sake and then a merging of methods with contents, the cognitive with the affective, and the traditional with the innovative. Let us be about this task!

APPENDICES

APPENDIX A

QUESTIONNAIRE

QUESTIONNAIRE

Name_____ Student Number_____

Classification (e.g. freshman, sophomore, etc.)_____

Permanent Address_____

1. What mathematics courses did you have in high school?

2. What mathematics courses have you had here at U-Mass
    or other universities?

3. What mathematics courses do you intend to take here
    at U-Mass? List Courses.

4. If you have not taken any university level mathematics
    courses to date, why have you chosen to take this methods
    course?

5. Are you a transfer student?

6. If the answer to the previous question is yes, please indicate what college you attended prior to attending U-Mass.

7. On a scale of 0-5 (0 being poor and 5 being excellent), how would you rate the following:

        a). Your mathematical ability   _____

        b). Your mathematical attitude   _____

        c). Your elementary mathe-
           matics instruction   _____

        d). Your high school Mathe-
           matics instruction   _____

        e). Your college mathe-
           matics instruction   _____

APPENDIX B

ATTITUDE – UNDERSTANDING
QUESTIONNAIRE

ATTITUDE - UNDERSTANDING
QUESTIONNAIRE

INSTRUCTIONS: The purpose of this questionnaire is to find out what changes, if any have occurred in your attitudes or understandings about mathematics, computers, programming, and teaching during your seven week "Mathematics and Programming" (MAP) Course.

If you feel that your attitude or understanding about a particular concept has <u>rapidly decreased</u> (RD) or <u>rapidly increased</u> (RI) since the beginning of the MAP Course, circle the graph which reflects your current attitude or understanding.

RD	SD	ID	NC	DI	SI	RI

RD	SD	ID	NC	DI	SI	RI
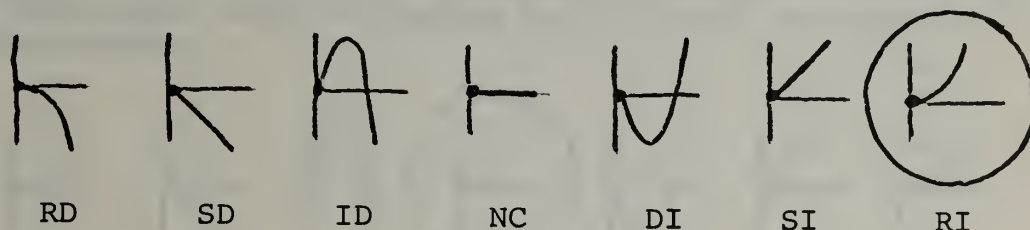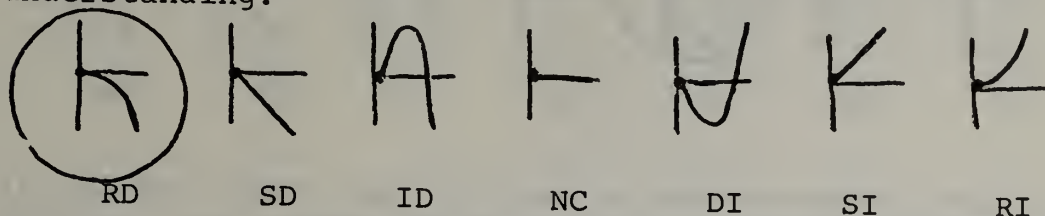
If you feel that your attitude or understanding about a particular concept has <u>steadily decreased</u> (SD) or <u>steadily increased</u> (SI) since the beginning of the MAP Course, circle the graph which reflects your current attitude or understanding.

RD	SD	ID	NC	DI	SI	RI

RD	SD	ID	NC	DI	SI	RI
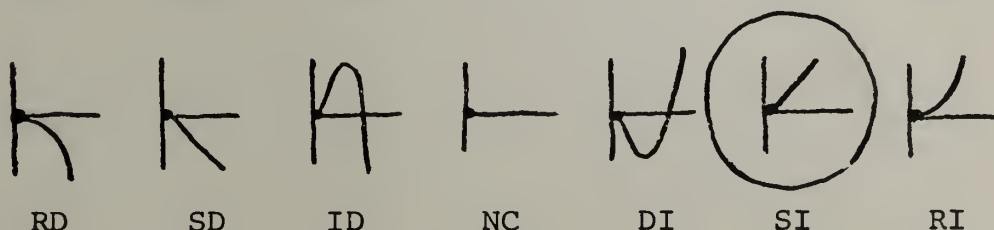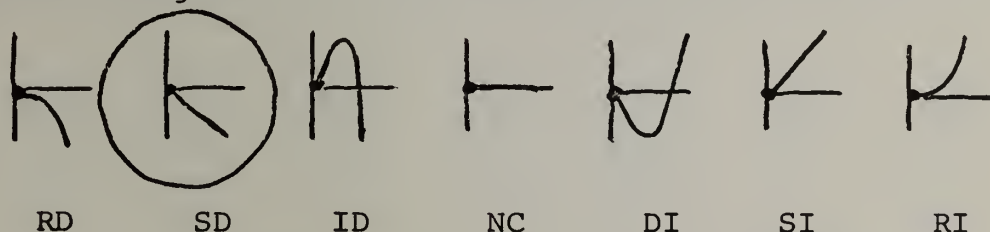
If you feel your attitude or understanding about a parti-
cular concept has <u>increased</u> then <u>suddenly decreased</u> (ID)
or was on the <u>decrease</u> then <u>suddenly increased (DI)</u> during
the MAP Course, circle the graph which reflects this
change.

RD   SD   ID   NC   DI   SI   RI

RD   SD   ID   NC   DI   SI   RI
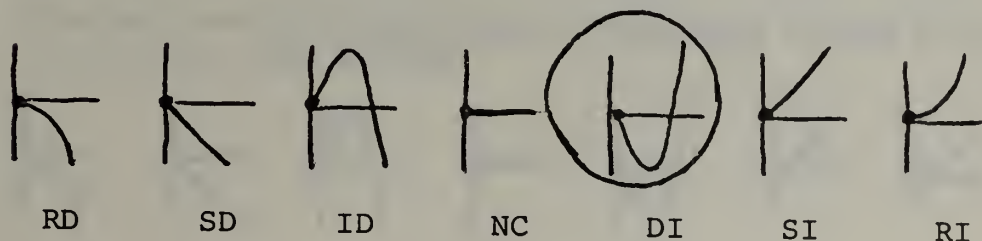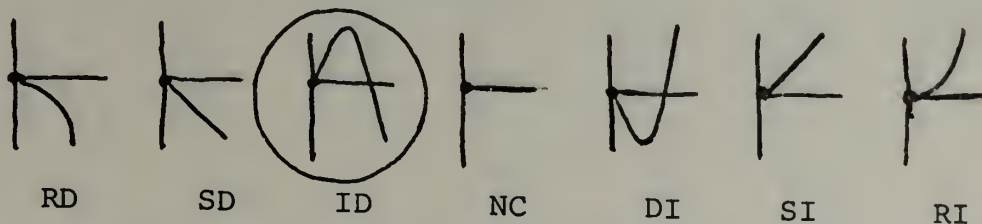
If you feel that your attitude or understanding has <u>not</u>
<u>changed</u> (NC) since the beginning of the MAP Course, cir-
cle the graph which indicates your current attitude or
understanding.
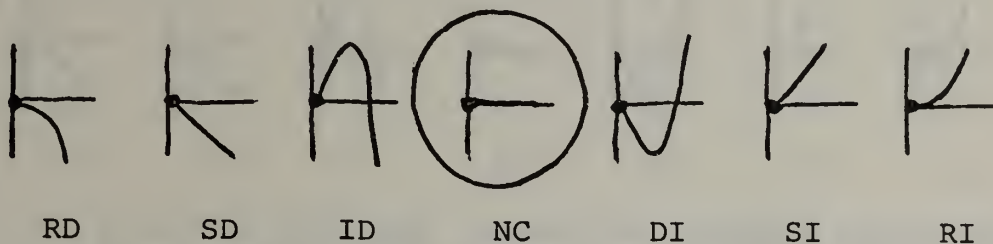
RD   SD   ID   NC   DI   SI   RI

ATTITUDE - UNDERSTANDING
QUESTIONNAIRE

1. How did your <u>attitude</u> toward <u>mathematics</u> change during this seven week course?
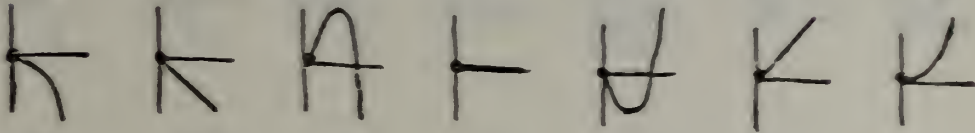
```
PD        SD        ID        NC        DI        SI        RI
```

2. How did your <u>attitude</u> toward <u>computers</u> change during this seven week course?

```
PD        SD        ID        NC        DI        SI        RI
```

3. How did your <u>attitude</u> toward <u>programming</u> change during this seven week course?

```
PD        SD        ID        NC        DI        SI        RI
```

4. How did your <u>attitude</u> toward the <u>teaching</u> <u>approaches</u> used in the MAP Course during this seven week course, namely:

      a). What was your attitude toward manual?

```
PD        SD        ID        NC        DI        SI        RI
```

b). What was your <u>attitude</u> toward <u>role-playing</u> (simulated sign-on, etc.)?

RD   SD   ID   NC   DI   SI   RI

c). What was your <u>attitude</u> toward "<u>Glass-Box programs</u>?

RD   SD   ID   NC   DI   SI   RI

d). What was your <u>attitude</u> toward <u>Games</u> (KNOWLEDGE-APL, APL500)?

RD   SD   ID   NC   DI   SI   RI

5. How did your <u>mathematics</u> <u>understanding</u> change during this seven week course?

RD   SD   ID   NC   DI   SI   RI

6. How did your <u>programming</u> understanding change during this seven week course?

RD   SD   ID   NC   DI   SI   RI

7. How did your <u>understanding</u> of <u>teaching techniques</u> change during this seven week course?

RD   SD   ID   NC   DI   SI   RI

APPENDIX C

PROGRAMMING ARTICULATION TEST

PROGRAMMING ARTICULATION TEST

Execute the APL commands below.   (Play computer and give the response you would expect the computer to display.) If a command is invalid give the appropriate error message.

Assumption: Each command is entered in a <u>clear workspace</u>.

ι4

2+3=5

1. _____

7. _____

|¯3.56

(3+(5×6))⌈8×4

2. _____

8. _____

⌊8.8

⌊3.5+5(7×2∧+5

3. _____

9. _____

-4+7

4*¯1

4. _____

10. _____

7+6.2-8×

4+8≤9|13*0

5. _____

11. _____

2|7+17-4×2

((+/ι5÷3),ι6

6. _____

12. _____

```
A       2   8   4   5   9   3.2

B       1   1   0   1   1   1

C       'WHEAT?'
```

      1↓⌽A   ⍳⍴B

13. _____   15. _____

         B/C   ∧/B

14. _____   16. _____

Program Examination

Each of the following defined functions is equivalent to
some primitive function. Evaluate each for a few arguments
and identify the corresponding primitive function.

```
        ∇Z←GUESS1 N                      ∇Z←X GUESS2 Y
   [1]  →(N≥0)/4              [1]  Z←X
   [2]  Z←0-N                 [2]  →3×X<Y
   [3]  →0                    [3]  Z←Y∇
   [4]  Z←N∇
```


17. _____   18. _____

```
        ∇Z←A MYSTERY B
[1]     Z←B,(~A∈B)/A∇
```

Which of the following set operations is embodied in the
above MYSTERY function:

a). UNION   b). INTERSECTION   c). COMPLEMENT

19. _____

Program Modification

Below is a function which rounds off numbers to nearest
tenth (e.g. 9.143 rounds off to 9.1 and 5.8625 rounds off
to 5.9)

```
        ∇ROUNDOFF N
[1]     N←N+.05
[2]     N←N+10
[3]     N←⌊N
[4]     N←N÷10
[5]     N
[6]     ∇
```

*WRITE YOUR PROGRAM
HERE*

Write a program which will round off numbers to the nearest
hundredth (e.g. 7.5873 rounds off to 7.59 and 2.31468
founds off to 2.31.

The following DIVISOR program is already in your workspace:

```
        ∇DIVISORS←DIVISOR NUMBER
[1]    ALL←ιNUMBER
[2]    REMAINDER←ALL|NUMBER
[3]    EVENLY←0=REMAINDER
[4]    DIVISORS←EVENLY/ALL∇
```

Use this DIVISOR program in your PRIME program which will print one (1) if a number is a prime and zero (0) if a number is not prime (e.g. If 7 is input 1 would be printed out, but if 9 is the input 0 would be printed out.)

WRITE YOUR PROGRAM HERE

A perfect number is one which is equal to the sum of all its divisors except itself. The first perfect number is 6

$$6 = 1 + 2 + 3$$

Use the DIVISOR program and write a PERFECTNUM program which will print out 1 if the number is perfect and 0 if the number is not perfect. (You may need to use the Drop ↓ function).

WRITE YOUR PROGRAM HERE

APPENDIX D

ANISA   CHECKLIST

ANISA CHECKLIST

Generally speaking, teachers may facilitate the development of creative competence in children by supporting and encouraging the following:

1. Deferring judgment (associated with the toleration of ambiguity);

2. Organize testing in a way that does not reinforce and emphasize convergence only;

3. Avoid punishing fantasy or daydreaming (though guiding a child to return from excess daydreaming may be necessary);

4. Emphasize process rather than product (although products must be evaluated, too);

5. Guide permissive play rather than punishing it;

6. Avoid intense argument and heavy criticism, particularly during the initial states of creative activity;

7. Encourage the production of quantity without regard to quality in the initial steps (it is important here to not give the child the idea that quantity is the sole criterion for determining excellence; quantity is needed in the initial stages of any creative effort);

8. Support any activity that involves differentiating attributes of any object, event, or idea. (This is a form of differentiating which is prerequisite to any creative act);

9. Encourage the use of metaphors;

10. Encourage speculation and guessing at appropriate times;

11. Avoid continual spirit of competition (this usually stresses convergence, and immediate judgments, both of which impair creativity);

12. Avoid routine approval of conformity behavior just for the sake of conforming;

13. Encourage children to select their own problems and projects (the teacher who demands that the children abide by her wishes on all matters is not encouraging a broad deployment of attention);

14. Avoid being authoritarian;

15. Introduce paradoxes to stimulate curiosity;

16. Avoid making premature closures on activities;

17. Use questions which confront the child with ambiguities or uncertainties (questions which cause the pupil to look at something from a different perspective; questions which require speculation and development of hunches; introduce mysteries and puzzling phenomena and ask questions that stimulate fantasies); and,

18. Encourage humor (laughter is a response to an unanticipated and therefore novel arrangement or integration of items or events. The first part of a joke, for instance, leads one to expect a particular conclusion or outcome; the punch line produces an unanticipated integration and we laugh. If someone doesn't grasp the novel integration we say that he doesn't "get" it. We sometimes laugh on such occasions, but it will be a faked laughter. Being around people with a good sense of humor facilitates creativity).

APPENDIX E

S.E.E.D. CHECKLIST

S.E.E.D. CHECKLIST

A checklist of questions to ask yourself on teaching
technique, method, etc.:

1. Do you start each of your classes with a conceptual
    review?

2. Does this review periodically cover all of the materi-
    al which you have worked on during the year?

3. How often do you circulate the class?
    a. Do you make the mistake of stopping at the desk
        of the pupil who is having trouble to provide
        individual instruction?
    b. Do you have the pupils hold their fingers under
        their answer in order that you can cover the class
        faster?
    c. Do you make individual contact with each pupil
        by speaking their name, touching their paper,
        crouching down beside them for a second, patting
        the arm, etc.?
    d. Do you mentally note which students, who normally
        do not participate in the verbal discussion, have
        correct answers?
    e. Do you use the knowledge in "d" to involve some
        of these people when the discussion begins again?

4. Do you Keep a log of each of your classes?
    a. Does it contain bits of dialogue to show the pupil's
        line of reasoning?
    b. Does it contain relevant comments about individual
        students?
    c. Does it contain comments on methodology that was
        successful or unsuccessful?
    d. Is there a comprehensive representation of what
        has been covered mathematically?
    e. When do you write these notes up--right after
        the class, in the evening of the same day, at the
        end of the week, etc.?
    f. Do you read the log before each class?
    g. Could another specialist read and understand the
        notes?

5. Do your pupils have a gesture of intellectual protest,
    such as waving their arms to indicate disagreement
    with what you or another pupil may have done?

a. Have you created an atmosphere in which students are willing, even in very small numbers, to disagree with the majority view held by the class and/or teacher?

b. Do you praise your students for catching your mistakes?

c. Do you occasionally, deliberately make mistakes to see if they can catch you?

d. Do your students argue with each other about mathematical ideas? How do you encourage this?

6. Do you have students give answers with their fingers in order that you can get a fast reading from the whole class?

a. Do you have students close their eyes before giving finger responses to minimize their copying one another?

b. Do you have students show their operational symbols, as well as numerical symbols, with their fingers?

c. Do the children have finger symbol which says, "I don't know"?

d. Do the children have a finger symbol which says, "It can't be done"?

e. Do you encourage your students to use the "I don't know" symbol and the "It can't be done" symbol?

7. How many times, each week, does a student in your class ask you a "meaningful" mathematical question?

a. What do you do to encourage children to ask questions?

b. Have you ever indicated to your class that a really good question deserves even more credit and praise than a good answer?

8. Do you put children's wrong answers on the board with a perfectly straight face?

a. What percent of the time are your children able to catch a straight-faced wrong answer?

b. Do you examine the children's technically incorrect answers to find out what they were thinking about?

c. Children's wrong answers usually result from one of the following. What percent of the wrong answers fall into each of the three categories?:

(1) Have they changed the set of assumptions (axiom system) underlying your question?

(2) Were they answering another question which they made up that was related to but different from your question?

(3) Was their answer seemingly random (it rarely is)?

d. When a wrong answer is based on a change in axiom system, do you treat this new system with respect and have the pupils solve problems using the new system? Do you give the new system the child's name?

e. If a child cannot answer a question, do you let him call upon another child to help him?

9. Is your class using a perfectly straight university mathematical vocabulary, or will your students have to unlearn the vocabulary you are using when they later study college preparatory mathematics?

a. Do you use Greek letters (it fascinates the children and gives them a sense of power)?

10. Do you give homework? How often?

a. Do you mark and return homework to the student?

b. Do you find that homework tends to turn the students off?

c. What do you do if the students do not turn their homework in?

d. Do you have each student who does not have his homework turn in a piece of paper indicating why?

e. Do your homework assignments contain at least a few questions that practically all the children can secceed with?

11. What are you doing about the children who are less involved or moving more slowly with their work?

a. Do you ever teach them something privately that the rest of the class does not know in order that they can "star" when you take it up at a later time?

b. Do you constantly search for less difficult questions to ask the slower student in order to provide them with success experience?

c. Have you had students whom you thought were "out of it" who have become involved again and are now functioning successfully?

d. How many children in your class have you worked with outside class time?

e. Have you had faster students tutor slower students?

12. If a child speaks so softly that you have trouble hearing him, do you call on individual students at distant points from the speaker to see if they heard what the speaker said?

a. When a child answers your question in a jumbled, inarticulate manner, do you put the implications of what he said on the board and let him discover that he didn't say what he meant?

b. Do you have children come to the board and ex-
plain things to the class?
c. Have you ever had a child teach the class for
as much as ten or fifteen minutes?

13. Are you careful to leave work that you have done up
on the blackboard long enough for students to refer
back to it as you continue the discussion?
a. Do you gradually alter a mathematical sentence,
which they understand, by erasing certain parts
of the sentence and replacing the erased parts
with other symbols that generalize or point up
subtleties?
b. If you write the next step of the problem separate-
ly, rather than altering it by the eraser method
of "a", do you write it beneath the preceding
step, or someplace else on the board?
c. If you write it beneath, do you use vertical ar-
rows to show how each item in the lower sentence
came from a corresponding item in the upper sen-
tence?
d. Would you be willing to bet that each child in
your class can see all of the material you have
written on the board? Have you ever sat in one
of their seats and looked at your own work on
the board?

14. Do you insist that students write intermediate steps
in order that you can have a better idea of how they
attain their answer?

BIBLIOGRAPHY

Allen, Dwight., and Bushell, D. D., ed. "Computers: Like
    Drugs and Atomic Power." Phi Delta Kappan, April,
    1968, p. 464.

Atkinson, Richard C., and Wilson, H.A., ed. Computer-As-
    sisted Instruction: A Book of Readings. New York,
    N.Y.: Academic Press, Inc., 1969.

Bartoli, Giovani; Bartolo, Luigi; and Spadavecchia, Vin-
    cenzo. "Design of a Simulator for Open Use of an
    APL Computer in Science Teaching." Paper presented
    at the APL V Conference, Toronto, Canada, May 15-18,
    1973.

Berry, Paul, et al. "APL and Insight: A Strategy for
    Teaching." Paper presented at the Conference on APL,
    Institut De Recherche D'Informatique et D'Automati-
    que, Paris, France, September 9-10, 1971.

Berry, Paul, et al. "APL and Insight: The Use of Programs
    to Represent Concepts in Teaching." IBM Technical
    Report, No. 320-3020, Philadelphia Scientific Center,
    (March, 1973).

Berry, Paul.; Iverson, Kenneth E.; and Falkoff, A.D. "Us-
    ing the Computer to Compute: A Direct but Negelected
    Approach to Teaching." IBM Technical Report, No.
    320-2988, New York Scientific Center, (May, 1970).

Bloom, Benjamin S., ed. Taxonomy of Educational Objectives
    Handbook I: Cognitive Domain and Handbook II:
    Affective Domain. New York, N.Y.: David McKay
    Company, Inc., 1956,1964.

Bruner, Jerome S. The Process of Education. Cambridge,
    Mass.: Harvard University Press, 1960.

Bundy, Robert F. "Computer-Assisted Instruction--Where Are
    We?" Phi Delta Kappan, April, 1968, p.428.

Calingaert, Peter. "Introduction to A Programming Language." Chicago, Ill.: Science Research Associates, 1968.

Carrol, Emma. "A Study of the Mathematical Understanding Possessed by Undergraduate Students Majoring in Elementary Education." Unpublished Ed.D dissertation, Wayne State University, 1961.

Committee on Guidelines of the Commission on Pre-Service Teacher Education of the NCTM. "Proposed Guidelines for the Preparation of Teachers of Mathematics." Paper presented at a meeting of the National Council of Teachers of Mathematics (NCTM), Chicago, Ill., April, 1972.

Conference Board of the Mathematical Sciences, Committee on Computer Education. Recommendations Regarding Computers in High School Education. Washington, D.C., (April, 1972).

Cooperative Mathematics Test. "Structure of the Number System." Princeton, N.J.: Educational Testing Service, 1964.

D'Augustine, Charles H., et al. New Dimensions in Mathematics, Teacher's Edition 5. New York, N.Y.: Harper Row, 1970.

Deans, Edwina, et al. Teacher's Annotated Edition of Learning Mathematics. New York, N.Y.: American Book Company, 1968.

de Charms, Richard. "Origin Pawns and Educational Practices." Washington University, (Mimegraphed.)

Dilley, Clyde A., Rucker, Walter E., and Jackson, Ann. Heath Elementary Mathematics 5. Lexington, Mass.: D.C. Heath and Company, 1972.

Duncan, Ernest, et al. Modern School Mathematics Structure and Use 5. Boston, Mass.: Houghton Mifflin Company, 1967.

Eicholz, Robert E., and O'daffer, Phares G. Elementary School Mathematics 5. 2nd ed. Menlo Park, Calif.: Addison-Wesley, 1968.

Feurzeig, Wallace, et al. "Programming Languages as a Conceptual Framework for Teaching Mathematics." An Introductory LOGO Teaching Sequence on Logic LOGO Reference Manual. Vol. I No. 2165, June 30, 1971.

Furth, Han G. Piaget for Teachers. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1970.

Gerschenkron, Alexander. Economic Backwardness in Historical Perspective. New York, N.Y.: Frederick A. Praeger, Inc., 1962.

Ginsburg, Herbert., and Opper, Sylvia. Piaget's Theory of Intellectual Development. Englewood Cliffs, N.J.: Prentice-Hall, 1969.

Halpern, Michael. "Permutations." Paper presented at the APL V Conference, Toronto, Canada, May 15-18, 1973.

Hammond, Allen J. "Computer-Assisted Instruction: Two Major Demonstrations." Science, CLXXVI (June, 1972), 1110-1112.

Hatfield, Larry L. "Computer Extended Problem Solving and Enquiry." Paper presented at the Annaul NCTM meeting, Anaheim, California, April, 1971.

Hatfield, Larry L. "Some Issues and Problems in Mathematics Teacher Education." Papers from the Forum on Mathematics Teacher Education NCTM. Paper presented at the National Council of Teachers of Mathematics meeting, Chicago, Ill., April, 1972.

Hilton, Peter J. "Recommendations on Course Content for the Training of Teachers of Mathematics." Papers from the Forum on Mathematics Teacher Education NCTM. Paper presented at the National Council of Teachers of Mathematics meeting, Chicago, Ill., April, 1972.

Holt, John. How Children Fail. New York, N.Y.: Pitman Publishing Corporation, 1964.

Hunkler, Richard. "A New Look at the Implementation of the CUPM Level I Recommendations." School Science and Mathematics, LXIX (May, 1961), 423-425.

Iverson, Kenneth E. A Programming Language. New York, N.Y.: John Wiley and Sons, Inc., 1962.

Iverson, Kenneth E. Algebra: An Algorithmic Treatment. Menlo Park, Calif.: Addison-Wesley Publishing Company, 1972.

Iverson, Kenneth E. "APL as an Analytic Notation." Paper presented at the APL V Conference, Toronto, Canada, May, 1973.

Iverson, Kenneth E. "APL in Exposition." IBM Technical Report, No. 320-3010, Philadelphia Scientific Center, (January, 1972).

Iverson, Kenneth E. Elementary Functions: An Algorithmic Treatment. Chicago, Ill.: Science Research Associates, Inc., 1966.

Iverson, Kenneth E. "Introducing APL to Teachers." IBM Technical Report, No. 320-3014, Philadelphia Scientific Center, (July, 1972).

Iverson, Kenneth E. "The Role of Computers in Teaching." Queen's paper in Pure and Applied Mathematics, No. 13, Queen's University, Kingston, Ontario, 1968.

Johntz, William F. "Innovation and the New Concern for the Disadvantaged." CTA Journal, (January, 1967), 5-6, 30-33.

Jones, Byrd, et al. Urban Education: The Hope Factor. Philadelphia, Pa.: W.B. Saunders and Company, 1972.

Jordon, Daniel., and Street, Donald. Releasing the Potentialities of the Child: A New Perspective on Child-Rearing, Day Care and Early Childhood Education. An unpublished report on the ANSIA MODEL at the University of Massachusetts in Amherst, (November, 1972).

Kurtz, Thomas E. (Principal Investigator). "The Computer as Pupil: The Dartmouth Secondary School Project." NSF GW-2246, Final Report, October, 1970.

LeBlanc, John. "Pedagogy in Elementary Mathematics Education--Time for a Change." The Arithmetic Teacher, XVII (November, 1970), 605-609.

Love, Barbara J. "Combatting Racism Through Teacher Training The Documentation of the Development of a Course in Survival Strategies." Unpublished Ed.D. dissertation, University of Massachusetts, 1972.

Luehrmann, Arthur W. "Large Scale CAI--The NSF Program." Sigcue Bulletin: Computer Uses in Education. Vol.6 No. 3, (June, 1972), 5-13.

Luehrmann, Arthur W. "Should the Computer Teach the Student or Vice Versa?" Paper presented at the Joint Computer Conference, 1972.

Mager, Robert F. Preparing Instructional Objectives. Belmont, Calif.: Fearon Publishers, 1962.

Masalski, William J. "The Design and Feasibility Study of the Mathematics Component of the Model Elementary Teacher Education Program." Unpublished Ed.D. dissertation University of Massachusetts, 1970.

Minsky, Marvin. "Form and Content in Computer Science." New Educational Technology, (Spring, 1973), 40-58.

Morton, Robert L., et al. Modern Mathematics through Discovery 5. Morriston, N.J.: Silver Burdett Company, 1970.

Mosteller, Frederick., and Moynihan, Daniel P. On Equality of Educational Opportunity. New York, N.Y.: Vintage Books, 1972.

Mueller, Francis J., ed. "Forum on Teacher Preparation." The Arithmetic Teacher, XVIII (April, 1971), 265-267.

McKee, Paul. A Primer for Parents. Boston: Houghton Mifflin Company, 1966.

McKee, Paul. Reading: A Program of Instruction for the Elementary School. Boston, Mass.: Houghton Mifflin Company, 1966.

National Council of Teachers of Mathematics. Topics in Mathematics for Elementary School Teachers. Twenty-ninth Yearbook. Washington: NCTM, 1964.

National Council of Teachers of Mathematics. More Topics in Mathematics for Elementary School Teachers. Thirtieth Yearbook. Washington: NCTM, 1968.

Nichols, Eugene D., et al. Elementary Mathematics Patterns and Structures 5. New York, N.Y.: Holt, Rinehart, and Winston, Inc. 1966.

O'Neill, Nena, and O'Neill George. _Open Marriage: A New Life Style for Couples._ New York, N.Y.: M. Evans and Company, Inc., 1972.

Orleans, Jacobs., and Wandt, Edwin. "The Understanding of Arithmetic Possessed by Teachers." _Elementary School Journal_, LIII (May, 1953), 501-507.

Osgood, Charles E.; Suci, George J.; and Tannebaum, Percy H. _The Measurement of Meaning._ Urbana, Ill.: University of Illinois Press, 1967.

Pakin, Sandra. _APL\360 Reference Manual._ 2nd ed. Chicago, Ill.: Science Research Associates, Inc. 1972.

Papert, Seymour. "A Computer Laboratory for Elementary School." _New Educational Technology_, (Spring, 1973), 26-30.

Papert, Seymour. "Teaching Children Thinking." _New Educational Technology_, (Spring, 1973), 1-6.

Papert, Seymour. "Teaching Children to be Mathematicians verses Teaching About Mathematics." _New Educational Technology_, (Spring, 1973), 7-20.

Payne, Joseph N., _et al._ _Elementary Mathematics 5._ 2nd ed. Newyork, N.Y.: Harcourt, Bruce, and World, Inc., 1968.

Peelle, Howard A. "A Study of Computer-Assisted Learning with Artificial Intelligence Games." Unplished Ph.D. dissertation at University of Massachusetts, 1971.

Peelle, Howard A. "Pygmalion's Computer." to appear in _Controveries in Education_. Philadelphia, Pa.: W.B. Saunders, 1974.

Peelle, Howard A. "Teaching Children Thinking via APL." Paper presented at the APL V Conference, Toronto, Canada, May 15-18, 1973.

Peelle, Howard A. "The Computer Glass Box: Teaching Children Concepts with A Programming Language." to be published by _Educational Technology_, (Spring, 1974).

Peelle, Howard A. "U-Programs." and "Mini U-Programs." Poughkeepsie, N.Y.: Shared Educational Computer System, Inc., (SECOS), 1971, 1972.

Penfield, Paul. "Proposed Notations and Implementation for Derivatives in APL." Paper presented at the APL V Conference, Toronto, Canada, May 15-18, 1973.

Pettigrew, Thomas F. "Racially Separate or Together?" Journal of Social Issues, XXV, No. 1 (1969), 43-69.

Piaget, Jean. Genetic Epistemology. Translated by Eleanor Duckworth. New York, N.Y.: W.W. Norton and Company, Inc., 1970.

Purkey, William. Self-Concept and School Achievement. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1970.

Rosenthal, Robert., and Jacobsen Lenore. Pygmalion in the Classroom. New York, N.Y.: Holt, Rinehart, and Winston, Inc., 1968.

Sage, Edwin. "Problem-Solving with the Computer." Newburport, Mass.: Entelek, Inc., 1969.

Sarason, Seymour B. The Culture of the School and the Problem of Change. Boston, Mass.: Allyn and Bacon, Inc. 1971.

Sparks, Jack. "Arithmetic Understanding Needed by Elementary School Teachers." The Arithmetic Teacher, (December, 1961), 395-403.

Suppes, Patrick. "Computer Technology and the Future of Education." Phi Delta Kappan, April, 1968, p. 422.

Wozencraft, Marian. "Even the Teacher Can't Do It!" Journal of Teacher Education, XI (September, 1960), 387-390.